1967-06

A computer graphics approach to non-linear circuit design.

Tynan, John Malcolm Calhoun

Monterey, California. U.S. Naval Postgraduate School

A COMPUTER GRAPHICS APPROACH TO NON-LINEAR CIRCUIT DESIGN

JOHN MALCOM CALHOUN TYNAN

A COMPUTER GRAPHICS APPROACH

TO NON-LINEAR CIRCUIT DESIGN

by

John Malcolm Calhoun Tynan
Lieutenant, Royal Canadian Navy
B.S., University of British Columbia, 1959

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ELECTRONICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1967

ABSTRACT

A COMPUTER GRAPHICS APPROACH

TO NON-LINEAR CIRCUIT DESIGN

This investigation is concerned with computer and programming requirements for on-line network design. A class of problem involving non-linear diode resistance networks was chosen. Such networks provide a useful form of general purpose function generator, here applied to the specific task of symbol generation. A procedure was implemented whereby the user "draws" the required symbol to be generated on a cathode ray tube display, and output is obtained in the form of a labeled circuit diagram of the synthesized network. The validity of the computer-graphic design program was demonstrated by construction and testing of a particular symbol generator.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF SYMBOLS

V	Voltage
I	Current
t(i)	Time i
t(if)	Fractional time defined as time i divided by total time n
RTOTAL(i)	Total function generator resistance connected in the circuit at time i
R(i)	Shunt resistor to be switched into the function generator circuit at time i
X(i)	Horizontal position coordinate at time i
Y(i)	Vertical position coordinate at time i
R(i1)	Branch i bias potentiometer resistance setting for potentiometer connected to ground
R(i2)	Branch i bias potentiometer resistance setting for potentiometer connected to bias supply
$\triangle F$	Small interval about some function F evaluated at an arbitrary point
Ro	Current sampling resistance used with function generators
k	Scale factor defined as the maximum coordinate input divided by maximum current in milliamperes of the display for which the symbol is being designed

ACKNOWLEDGEMENT

I would like to acknowledge the assistance and encouragement of Professor Mitchell Cotton, who initially conceived the idea for this work. His advice on several technical matters was invaluable and much appreciated.

CHAPTER I

INTRODUCTION TO THE GENERAL DESIGN PROBLEM

1. OBJECTIVES OF THE APPROACH TO COMPUTER DESIGN

It is the aim of this work to study the modern technique of engineering design by digital computer using graphical methods of data display. Details of the methods required for both input and output data display will be investigated. For many problems graphical displays are the most direct communication link between man and machine. In order to illustrate these techniques, a problem in nonlinear linear network synthesis was chosen. This problem involves the synthesis of the circuitry to produce the waveforms to generate a symbol. Because of the nonlinear nature of the waveforms, the synthesis involves an iterative numerical solution. A system of graphical input and output data display was devised for this design problem. It is hoped that this study will yield insight into a general approach to design using these methods.

2. BASIC FUNCTIONS OF THE COMPUTER AIDED DESIGN SYSTEM

In many cases the choice of the specific method used in a particular facet of the example design problem was dictated by the practical limitations of the computer and its peripheral equipment. Regardless of these limitations, the system operates satisfactorily and serves well to demonstrate the salient features of this approach to design. As a particular example, the computer design of the letter "L" will be studied. The first phase of the design procedure is to input the data into the computer. The primary input source is the track-ball which may be positioned in X and Y coordinates. The track-ball analogue position is

converted to nine bit ones complement digital position coordinates. The data are entered by drawing a piecewise linear sketch of the letter "L" on the display using the track-ball. The sequence of strokes used to sketch the letter will be the same sequence of strokes produced by the designed circuit in displaying the letter. The track-ball data are converted to a twenty-two bit format as it is entered into the data tables as illustrated in figure 1. The secondary source of data would be data produced as above and prestored on magnetic tape. The next step in the design is to select the compute operation. There are two arguments which may be altered to suit the specific design requirements. The first is the voltage in millivolts which is used in the sweep supply for the designed symbol generator circuit. The second is the ratio of input data units, describing symbol size, to the corresponding output of the designed circuit in milliamperes. Saying this another way, the vertical leg of the letter "L" has a length in input units. The designed circuit must produce a specified number of milliamperes to produce the desired height. The relationship of these two arguments in the design procedure is more fully covered in Chapter III. After selecting the compute operation the computer does the proper calculations and automatically compiles the output display file. In the case of the letter "L", there are two piecewise linear regions. The circuitry therefore consists of two series resistance diode legs connected in parallel in both the "X" and "Y" circuits. Only the "X" or "Y" circuit is displayed at one time. Figure 1 shows the format of the "Y" display. It is noted that the resistances shown in each leg act both as voltage dividers to produce the proper bias voltage for the diode and as circuit resistance which is switched into the circuit by the diode. Figure 2 shows how

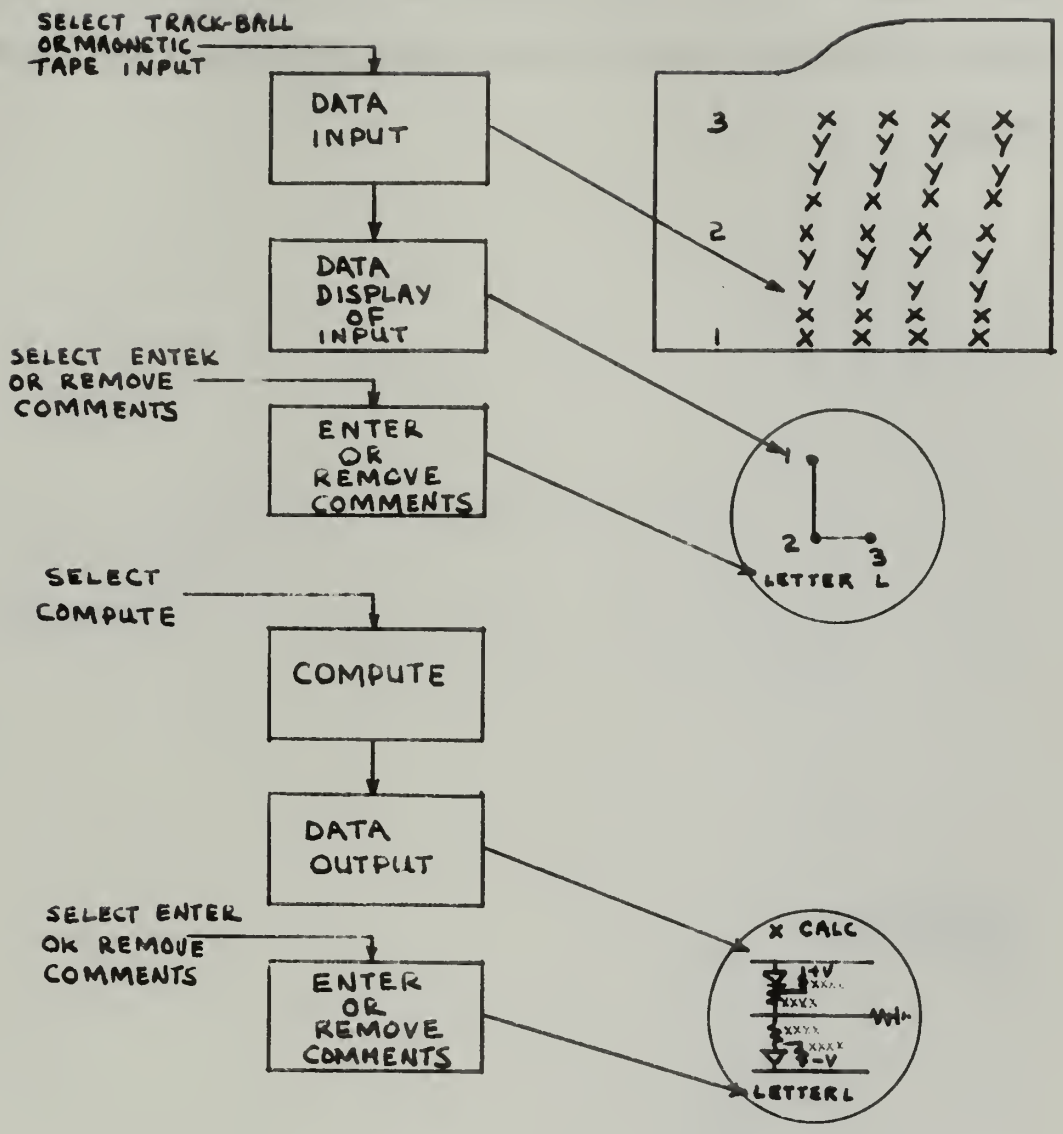

FIGURE 1

FUNCTIONAL BLOCK DIAGRAM OF SEQUENCE OF COMPUTER DESIGN OPERATIONS

the "X" and "Y" circuit output waveforms form the letter "L". The theory of how these circuits produce these waveforms is left to Chapter II.

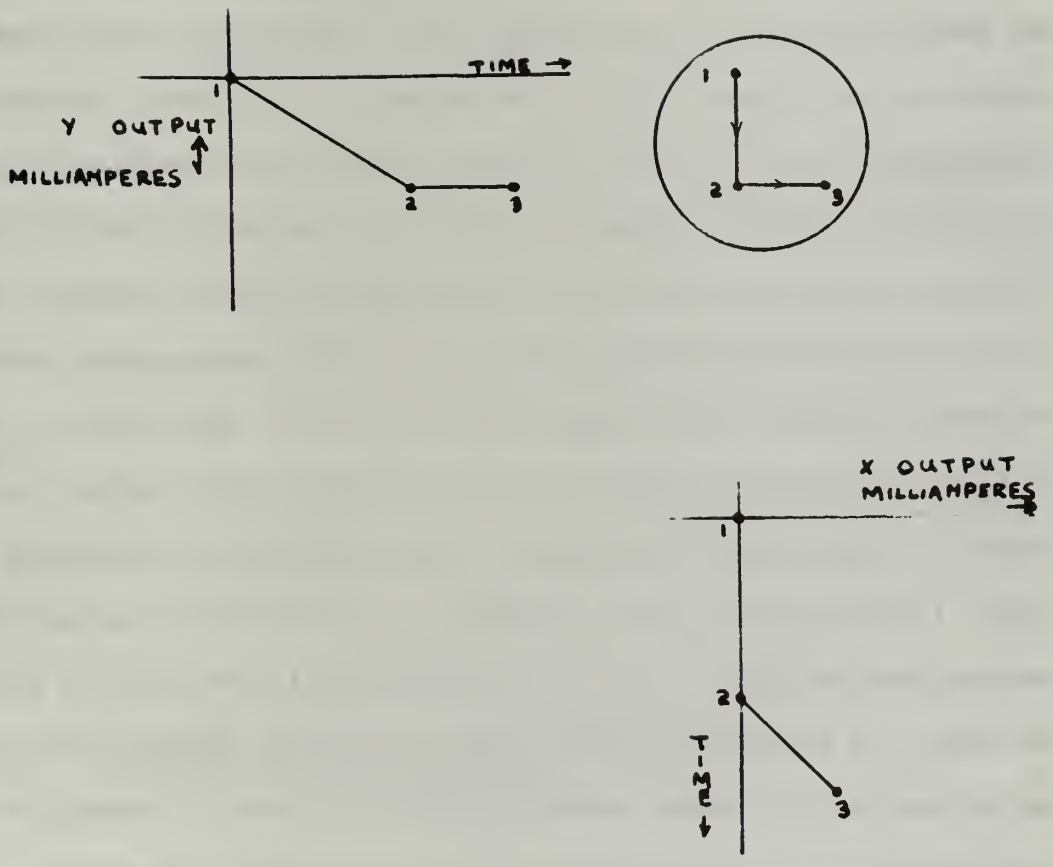

FIGURE 2

ILLUSTRATION OF HOW CIRCUIT WAVEFORMS
PRODUCE THE LETTER "L"

CHAPTER II

COMPUTER DESIGN APPLIED TO THE DESIGN OF SYMBOL GENERATORS

1. FUNCTION GENERATORS AND THEIR DESIGN BY ALGORITHMIC PROCEDURES

As indicated in Chapter I, the design problem is solved using an algorithm which results in an iterative computational process. The specific design being considered is that of symbol generators. The symbol generators to be considered are function generators which consist of resistances and biased diodes. The method by which these function generators may be used to obtain piecewise linear approximations to symbols will now be described. Figure 3 illustrates the generation of a piecewise linear monotonic function using a parallel bank of resistors which are switched into the circuit by means of single-pole, single-throw switches which are closed sequentially with time. Note that the circuit of figure 3 produces a monotonically increasing current output function. Figure 4 illustrates the generation of a monotonically decreasing function. It is possible by the summation of a monotone increasing and a monotone decreasing function to produce a function which is no longer monotone. As a practical aspect, the single-pole, single-throw switches may be replaced by diodes biased at different levels of voltage so that they switch at various times after the input sweep is commenced. Since the input is linear with time this corresponds to the diodes switching at various input voltages. The current outputs are added in a summing network. At any instant of time the value of the output current slope from the function generator is the product of the voltage slope divided by the total parallel resistance which is in the circuit at that time.

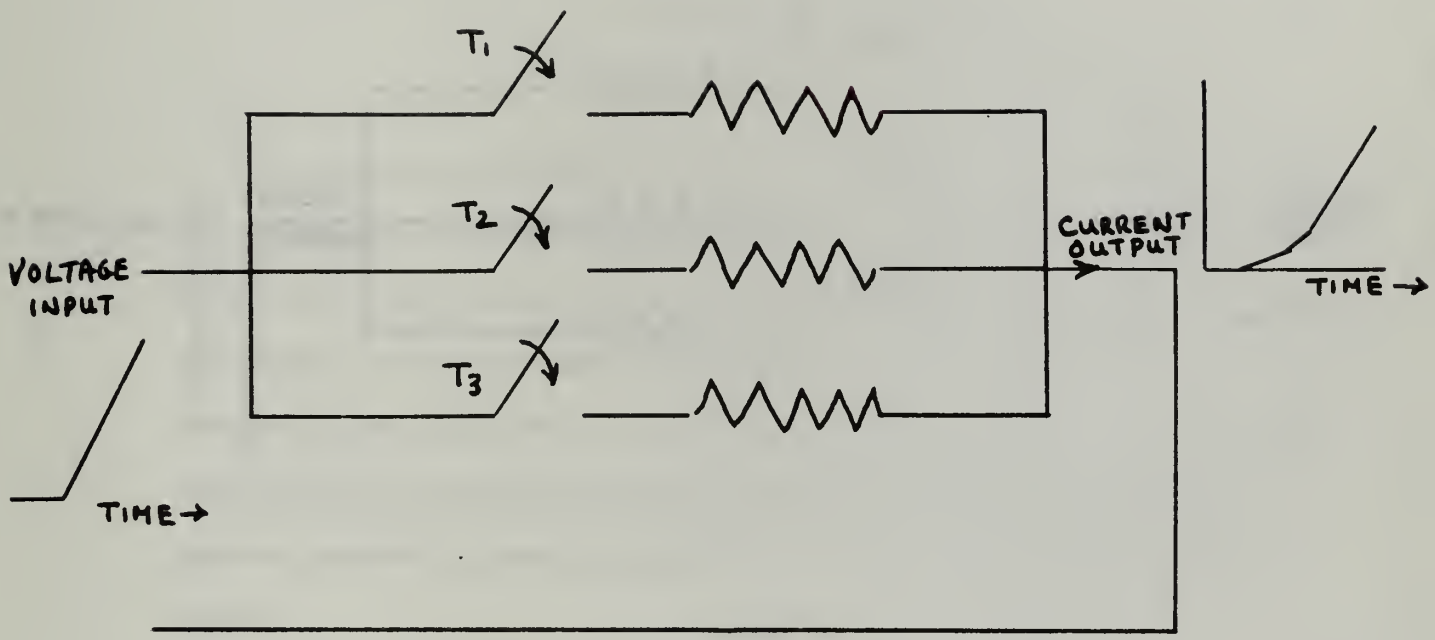

FIGURE 3

A CIRCUIT DIAGRAM OF A FUNCTION GENERATOR WHICH PRODUCES A MONOTONICALLY INCREASING FUNCTION OF TIME

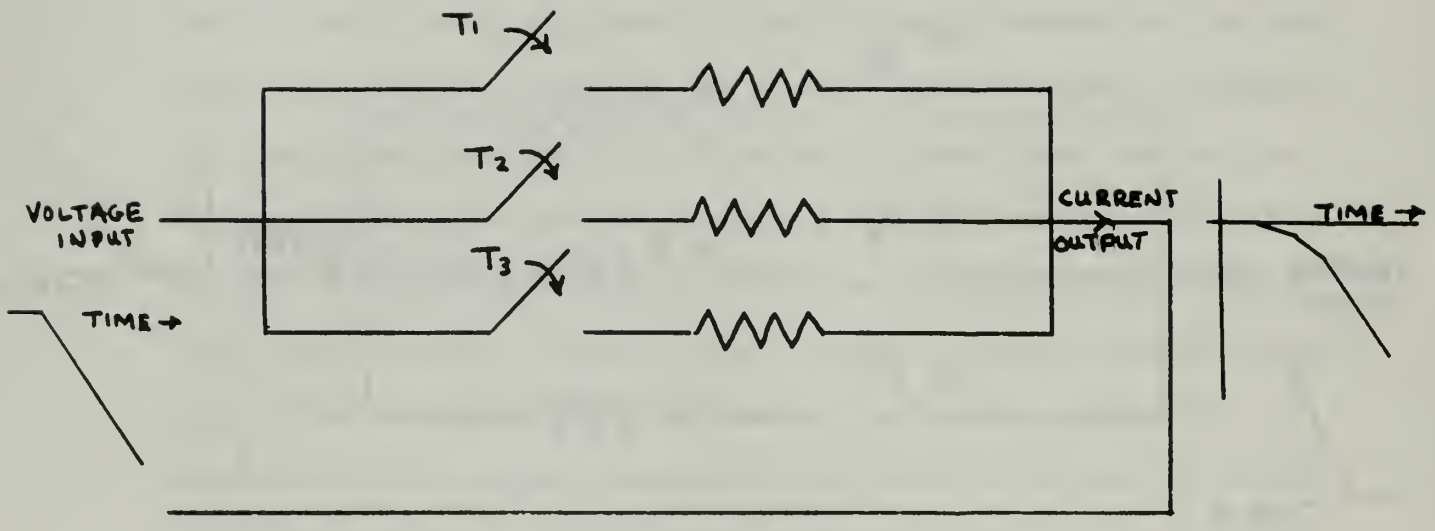

FIGURE 4

CIRCUIT DIAGRAM OF A FUNCTION GENERATOR
WHICH PRODUCES A MONOTONICALLY DECREASING
FUNCTION OF TIME

This gives a positive increase of current with time for the case of figure 4. Since the outputs of two function generators are summed together, the resulting current slope is an algebraic addition of the two current slopes. Now for any case, the total network resistance is defined as the voltage slope $\frac{dv}{dt}$ divided by the current slope $\frac{dI}{dt}$ which may be algebraically positive or negative.

$$RTOTAL = \frac{\frac{dv}{dt}}{\frac{dI}{dt}} \tag{1}$$

In order to determine the resistance required to be added to a function generator to produce a new specified slope (new total network resistance), it is necessary to take the present total network resistance and compute the resistance to be added in parallel to produce the new desired network resistance. If the resistance turns out to be negative, this means the resistance is to be paralleled with the resistors in the monotone-decreasing function generator. This is shown symbolically as follows:

$$RTOTAL(2) = \frac{RTOTAL(1)\ R(2)}{RTOTAL(1) + R(2)}$$

where RTOTAL(2) is the parallel resistance of RTOTAL(1) and R(2)

Hence $R(2) = \frac{RTOTAL(1)\ RTOTAL(2)}{RTOTAL(1)-RTOTAL(2)}$

This may be written in the form of an algorithm:

$$R(i+1) = \frac{RTOTAL(i)\ RTOTAL(i+1)}{RTOTAL(i)-RTOTAL(i+1)} \tag{2}$$

with R(1) = RTOTAL(1)

2. DESIGN OF SYMBOL GENERATORS USING FUNCTION GENERATORS

Symbols may be displayed on a cathode ray oscilloscope by applying the correct wave forms to the two pairs of orthogonal plates. These wave forms must be designed so that they move the tube electron beam at a constant rate, as it sweeps out the character, so that a constant intensity figure is painted on the tube face. This is ensured by arranging that the wave form on each of the two orthogonal plate pairs satisfies the restriction that it change its amplitude at a constant rate. Also, since the wave forms must be in time phase, switching must occur simultaneously in both the function generators at each change in slope of the symbol being generated. This means that the individual plate wave forms may be designed separately against a dummy time variable. The latter is derived from the fraction of the total distance traveled between the discontinuous slope changes of the symbol generated. This distance is obtained from the input data which is in the form of two coordinates defined for each discontinuity in the symbol to be generated. Let these coordinates be defined X and Y. Considering X only, the change in t in time units (distance), between adjacent coordinates denoted X(i) and X(i+1) respectively, is calculated. Each of these individual time intervals is summed up to the i+1th coordinate to give t(i+1). If there are n coordinate points, the last time is t(n+1). The fractional time is given by equation three.

$$t(if) = \frac{t(i)}{t(n+1)} \qquad (3)$$

where $t(1) = 0$

Now to compute resistance RTOTAL(i), it is necessary to know what the current slope is during the i+1th interval. The X coordinates are proportional to the current required from the X function generator as

defined by equation four.

$$I(i) = \frac{(X(i))}{k} \tag{4}$$

where k is the scale factor in units of input X per milliampere of current.

Thus it is seen the current slope is given by equation five.

$$\text{Current rate milliamperes/sec.} = \frac{\frac{1}{k}\,(X(i+1)-X(i))}{t(i+1)-t(i)} \tag{5}$$

Now t(nf), derived from equation three for the index n, is 1 and letting Vmax be the maximum input voltage, the input voltage rate in millivolts/second is Vmax. By substituting these values in equation 1, one obtains the expression given in equation 6.

$$RTOTAL(i+1) = \frac{KVmax(t(i+1)-T(i))}{(X(i+1)-X(i))} \tag{6}$$

The index i in equation six goes from 1 to n. In the special case where i is unity, the value of t(1) is zero. In the case where the resistance diode network is biased by some series bias voltage which opposes the input sweep voltage, the effective branch resistance is higher than that given in equation 6. A correction factor may be added to Vmax to account for this. Since the bias voltage $E(i) = \frac{Vmaxt(i)}{t(n+1)}$, the correct RTOTAL is computed using equation 6a.

$$RTOTAL(i+1) \quad \frac{k(t(i+1)-t(i))\ Vmax(1-\frac{t(i)}{t(n+1)})}{(X(i+1)-X(i)} \tag{6a}$$

Equation 6a simplifies to equation 6 in the case that the bias voltage is zero. This completes the algorithm. The order of computation is equation 6a then equation 2. The scale factor used on the input data is defined in equation 4. It is the ratio of the maximum input X to the maximum current output from the function generator. The latter value is

arrived at from power dissipation considerations while the former is determined by the input device.

CHAPTER III

THE PRACTICAL IMPLEMENTATION OF THE SYMBOL GENERATOR DESIGN ALGORITHMS

1. SYMBOL GENERATOR DESIGN

As mentioned in paragraph 1 of Chapter II, the switching devices used in the function generators are biased diodes. Figure 5 indicates the practical method of obtaining the required resistance diode network. The equivalent circuit of this network is illustrated in figure 6. Initially it is assumed that the diode D(i) and the power supply Vmax, shown in figure 5, are both ideal in nature. This being the case, the Voltage E at point E of figure 6 is given by equation 7 while the resistance R(i), looking into the same point, is given by equation 8.

$$E(i) = \frac{R(i1)\ Vmax}{R(i1)+R(i2)} \tag{7}$$

$$R(i) = \frac{R(i1)\ R(i2)}{R(i1)+R(i2)} \tag{8}$$

Solving these equations for Ri1 and Ri2 the relationships given in equations 9 and 10 are obtained:

$$R(i1) = \frac{Vmax\ R(i)}{Vmax-E(i)} \tag{9}$$

$$R(i2) = \frac{Vmax\ R(i)}{E(i)} \tag{10}$$

In actual fact the diode D(i) of figure 5 has some resistance even when forward biased by several milliamperes. Figure 7 shows the value of forward current plotted against bias voltage for a typical silicon diode. The effect of this resistance will be discussed later. Additionally, the resistor Ro of figure 5 affects the bias of diode D(i). The total parallel resistance of all R(i), as illustrated in figure 6, was constrained to be greater than one hundred ohms in the designed symbol

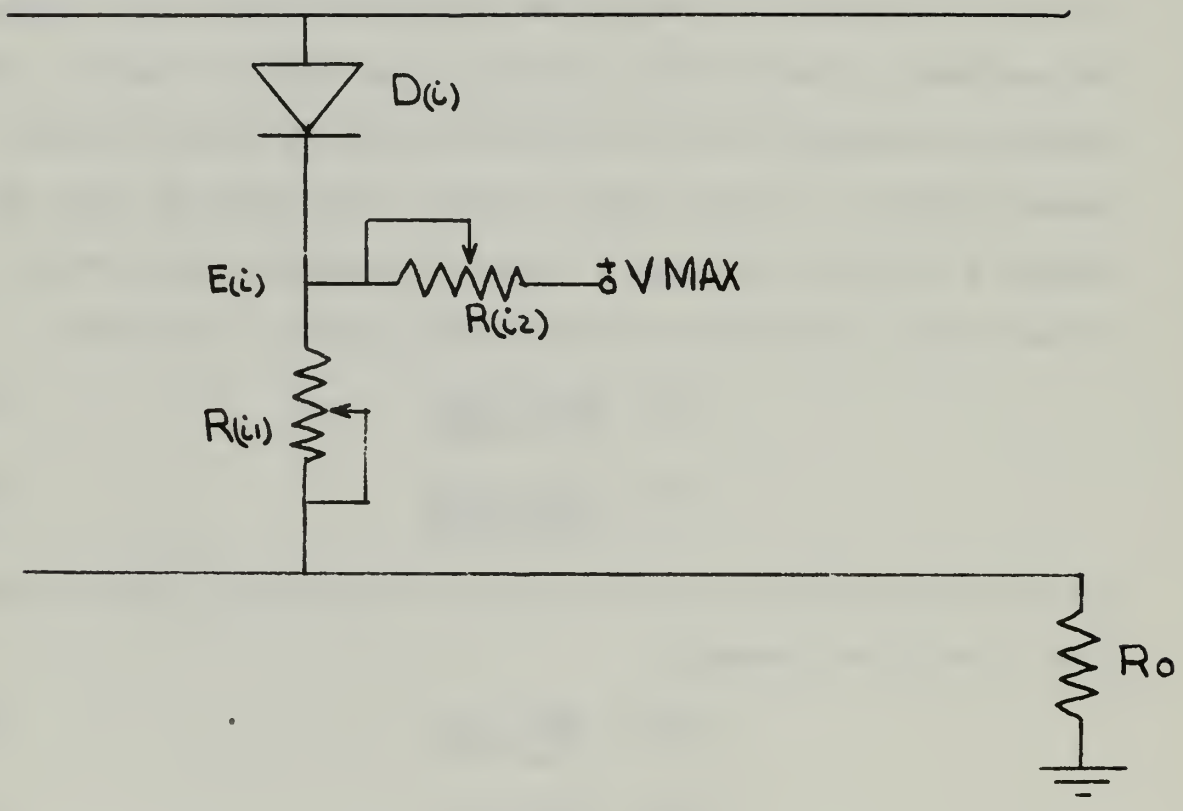

FIGURE 5

A TYPICAL SECTION OF A FUNCTION GENERATOR
WITH A MONOTONICALLY INCREASING OUTPUT
SHOWING DIODE BIASING ARRANGEMENT

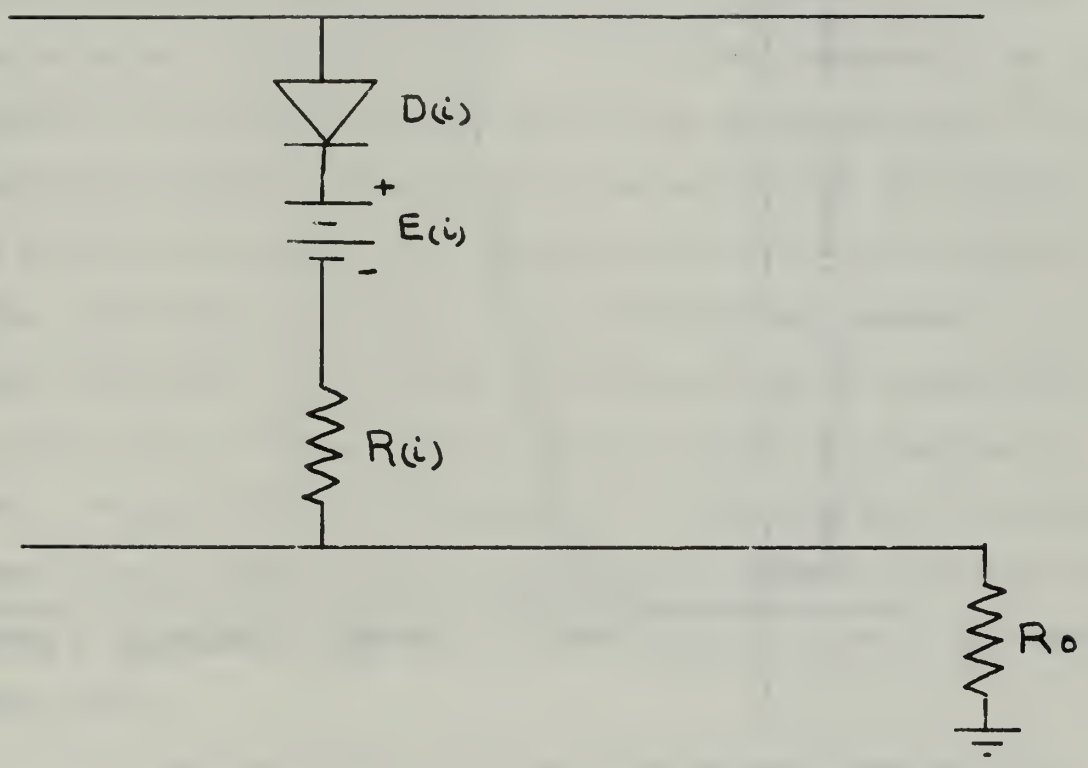

FIGURE 6

EQUIVALENT CIRCUIT OF A TYPICAL SECTION OF A FUNCTION GENERATOR THAT PRODUCES A MONOTONICALLY INCREASING OUTPUT FUNCTION

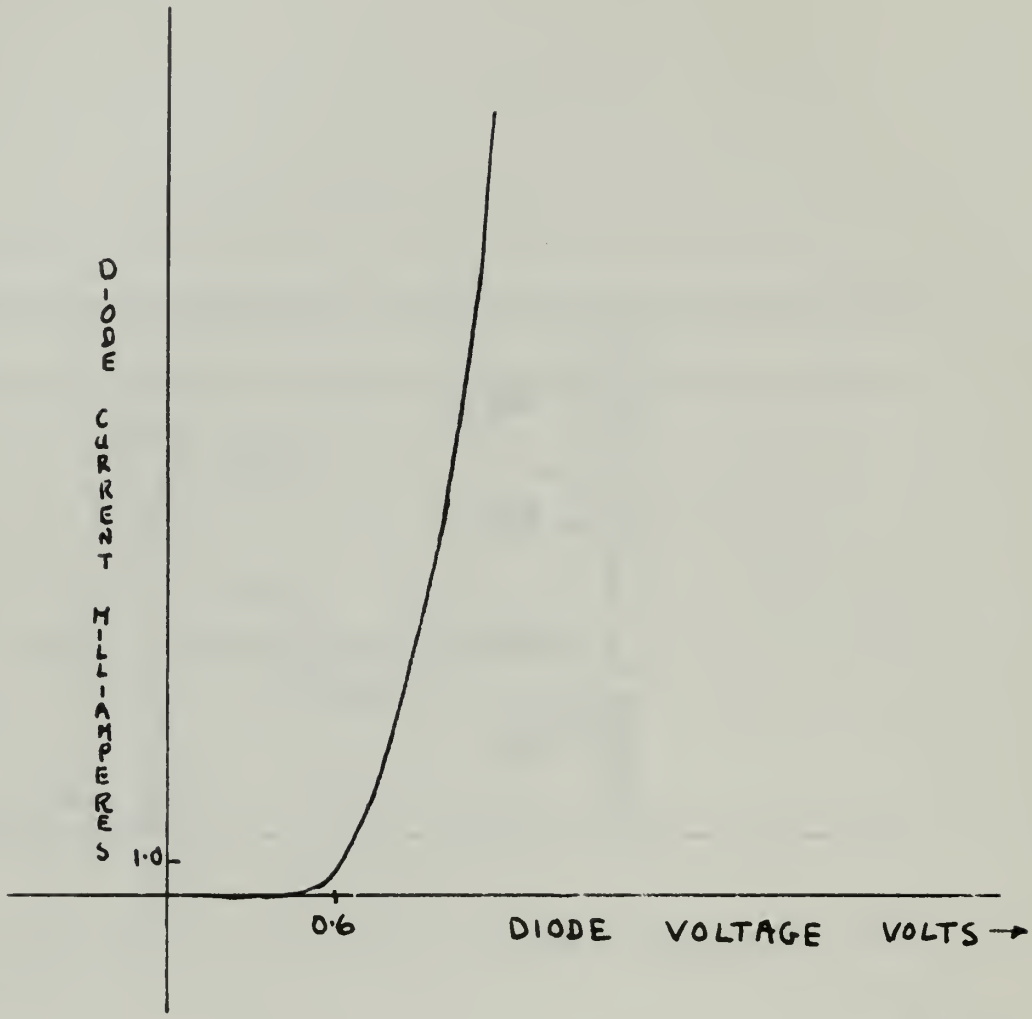

FIGURE 7

A CHARACTERISTIC CURVE FOR A
TYPICAL SILICON DIODE

generators. The value of Ro could add appreciable error to the bias level when the circuit resistance is low. A related error is found in the internal impedance of the supply Vmax. These latter errors must be compensated for by subtracting a compensating resistance value from the computed R(i2) to give the corrected potentiometer setting. It is noted that the battery internal impedance would be shunted by the remaining series values of R(i1) and R(i2). If the internal impedance of the bias supply is large, these resistors will have an appreciable effect. However if the internal impedance of the bias supply, plus Ro of figure 5, is less than twenty ohms, the correction factor will be very nearly 20 ohms. Returning to the question of the diode forward impedance, it is seen from figure 6 that this must be subtracted from the computed R(i) to give correct results. However, the bias voltage is a function of time. To remove this time dependence, it is necessary that the value of Vmax be large so that the diode resistance is negligibly small after the diode is switched. Values of 20 volts or more were found to give excellent results.

It is seen from figure 8 that the load resistor R must be large enough so that as VTOTAL changes, the loadline moves over a linear portion of the diode characteristic and produces a current which is linearly related to the current which would flow if the resistance were there alone. However, the resistor must be small enough to ensure fast switching time, as the rate of change of bias current is proportional to the rate of change of VTOTAL and inversely proportional to the load resistor R. This is the chief reason for constraining R to be between 100 and 10,000 ohms.

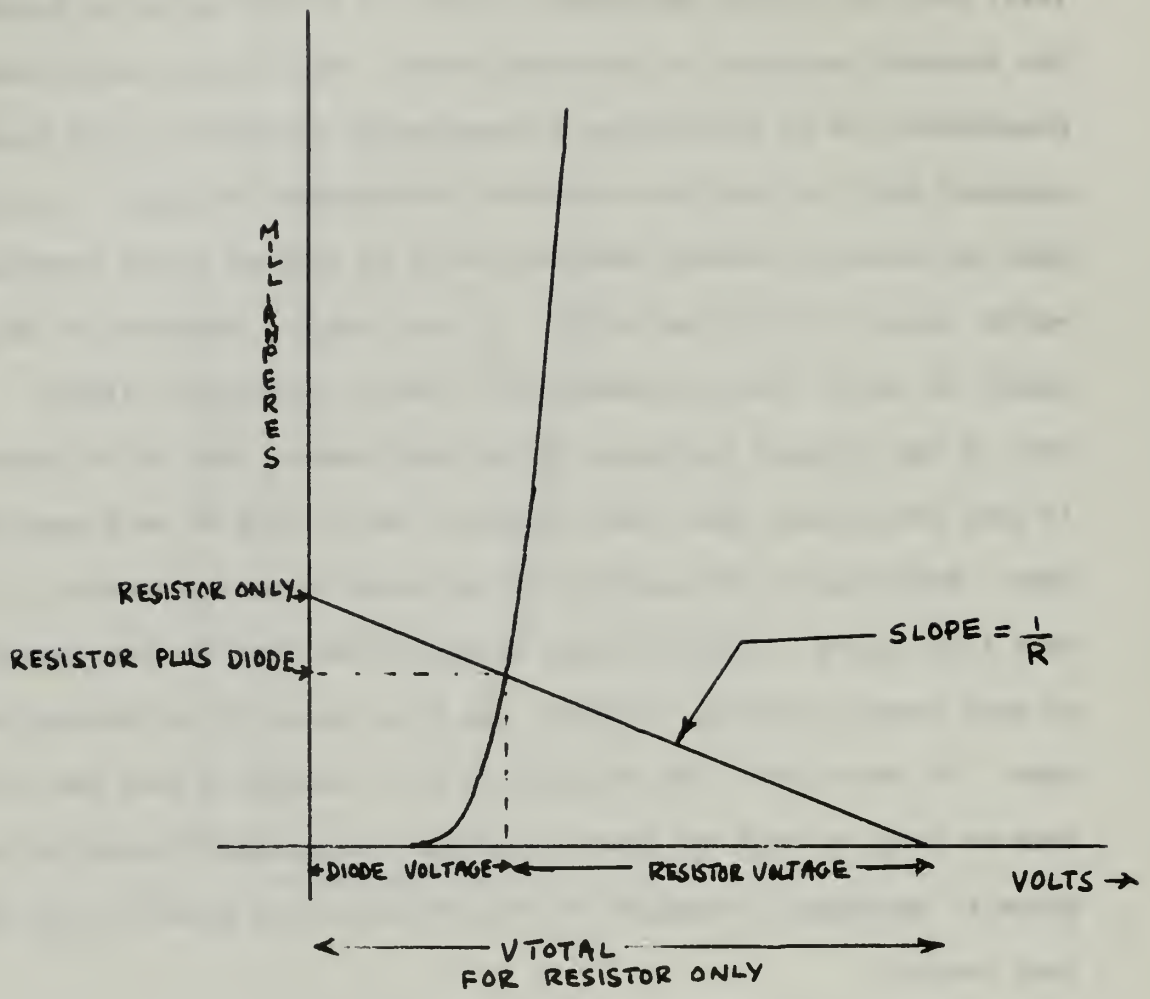

FIGURE 8

VOLTAGE CURRENT RELATIONSHIPS FOR A SERIES
RESISTOR DIODE NETWORK COMPARED WITH
THOSE FOR A RESISTOR ONLY

2. SYMBOL GENERATOR SENSITIVITY

It is of importance that the symbol generators not only perform their specified task in the best possible manner, but that they do so with good reliability. The reliability aspect mainly refers to the change in component parameters with time. The voltages fed to the symbol generators may be electronically regulated. The components themselves are sufficiently simple that they should exhibit a low failure rate. However, there are factors which would cause circuit resistance values to change with age. The following analysis will determine the seriousness of this problem.

From equation 1 it is seen that

$$\frac{dI}{dt} = \frac{dV}{\frac{dt}{RTOTAL}}$$

where RTOTAL at any time is made up of a number of parallel resistors R. That is:

$$\frac{1}{RTOTAL} = \frac{1}{R1} + \frac{1}{R2} + \cdots \frac{1}{R(i)}$$

Hence $$\frac{dI}{dt} = \frac{dv}{dt}\left(\frac{1}{R1} + \frac{1}{R2} + \cdots \frac{1}{R(i)}\right)$$

Now $$\frac{\frac{\partial (dI)}{\partial t (dt)}}{\partial R\ (i)} = \frac{dV}{dt}\frac{1}{R(i)^2}$$

$$-\frac{dV}{dt}\frac{1}{R(i)^2} = -\frac{dI}{dt}\frac{RTOTAL}{R(i)^2}$$

From the last two expressions equation 11 is obtained:

$$\frac{-\triangle \frac{dI}{dt}}{\frac{dI}{dt}} = -\frac{RTOTAL}{R(i)}\frac{(\triangle R(i))}{(R(i))} \tag{11}$$

Similarly, differentiating the expression for $\frac{1}{RTOTAL}$ with respect to R(i):

$$-\frac{1}{(RTOTAL)^2}\frac{\partial RTOTAL}{\partial R(i)} = -\frac{1}{R(i)^2}$$

$$\frac{\triangle RTOTAL}{RTOTAL} = \frac{RTOTAL}{R(i)}\frac{\triangle R(i)}{R(i)} \qquad (12)$$

Thus it is seen from equations 11 and 12 that the value of RTOTAL, which determines the output current, and $\frac{dI}{dt}$ have sensitivities proportional to the sensitivity of R(i) by the proportionality constant $\frac{RTOTAL}{R(i)}$. This proportionality constant is less than unity since R(i) is one of the parallel resistances comprising RTOTAL. Hence the circuit will exhibit good circuit stability. Furthermore, by setting these resistance values on potentiometers using a bridge, the symbol distortion due to inaccurate values of R(i) will be less than the error of the bridge which can be very small.

3. SOFTWARE STRUCTURE

The following paragraphs describe the computer software used in the particular design problem of Chapter II. Figure 9 shows the software used in the complete computer designs system.

The primary data source is the track-ball which is used in conjunction with the data input routine. The latter converts track-ball data to the fixed computer-word format and enters it in the data tables. Once the data are entered in the data table the data display subroutine feeds the information to the data display file which is displayed by the data display subroutine. Control is generally left in the track-ball driver routine which refers all action requests to the console control routine. The secondary source of data is magnetic tape which is operated as a service routine to fill the data table and display file. Comments may be added to or removed from the data display file. These operations

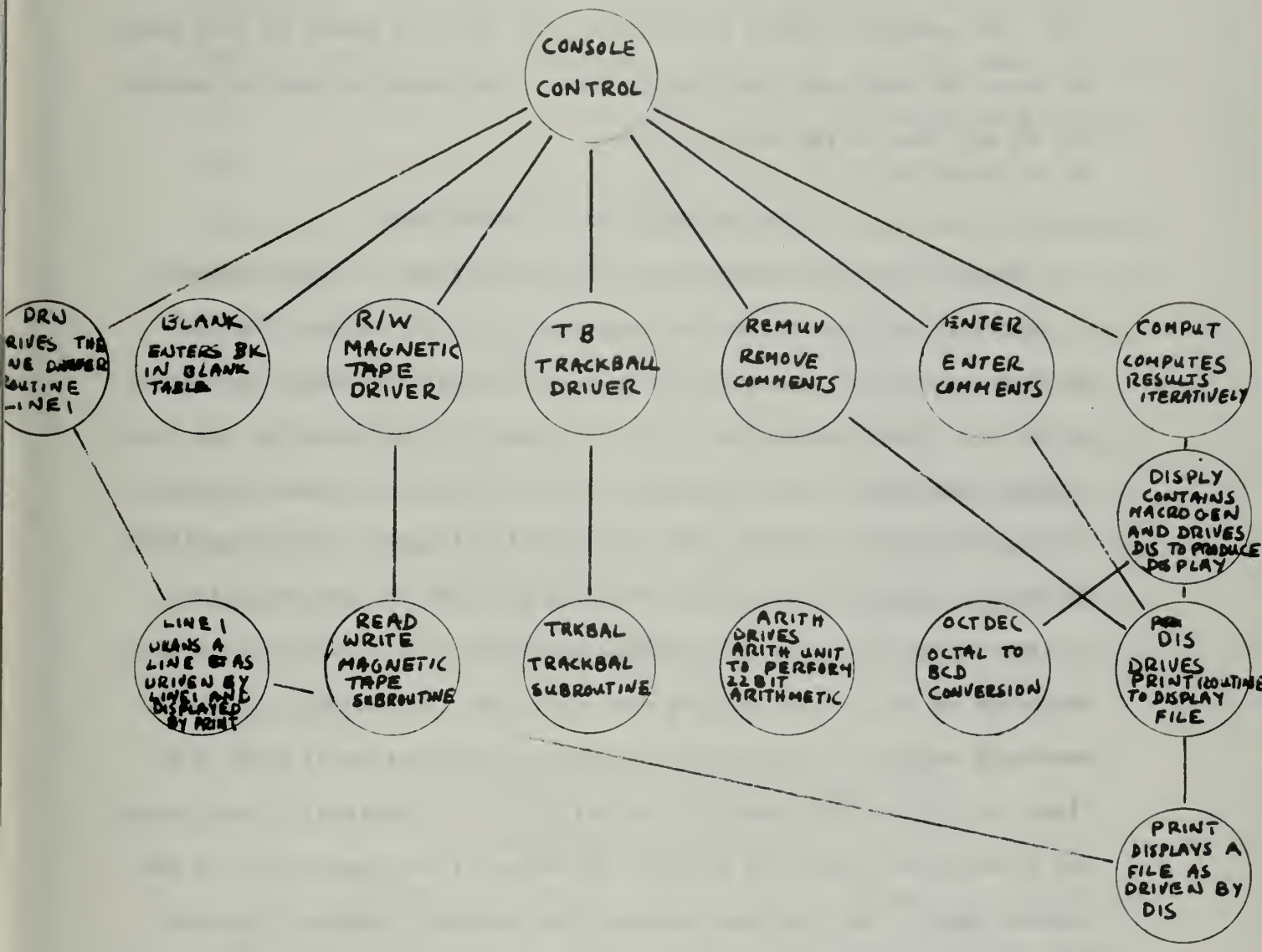

FIGURE 9

SOFTWARE USED IN THE COMPUTER DESIGN
OF SYMBOL GENERATORS

bypass the data display table and so have no effect on the design process. The subroutine which enters data into the data tables is also used to erase the data table and display file. This operation may be carried out at any time in the design process.

4. DETAILED OPERATION OF SUBROUTINES

Most of the subroutines need little explanation as their sequence of operation is flow charted in figures 10 to 23 inclusive. The subroutine comput and its associated subroutine disply are shown in figures 18 and 19. These subroutines do the arithmetic computations of the algorithmic equations 2, 6a, 7, and 8. The calculations are done in twenty-two bit fixed-point format. The fixed point arithmetic was accomplished by using a floating scale factor ensuring the numbers were manipulated to stay within the computer's maximum precision size. Calculations were performed so as to leave scaling down until the last in order to avoid round-off errors. Considering the square-root operation of block E of figure 18, it is noted that this calculation is a completely fixed point. The square-root subroutine computes the value of the square-root to the nearest digit. The computer computational accuracy compares favorably with the input accuracy which is point two percent. Appendix III gives details on methods used to maintain computational accuracy. The computed resistance values in the computer program of Appendix II were constrained to lie between 64 and 8192 ohms. These numbers cover the range of 100 to 10,000 ohms in octal. The reason for the restricted range is seen by considering figure 8, in which it is seen that R, which corresponds to a R(i) of equation 1, acts as the load resistance for a diode. Load resistors much larger than the upper value of 8192 ohms cause the diode to switch too slowly, while the lower value is used because of power

dissipation considerations. The next subroutine to be considered is disply which is the display driving subroutine. The flow chart for disply is illustrated in figure 19. This subroutine assembles a display file. Display outlines are stored in memory and transferred to the display file. These outlines have internal positions where information is stored before it is transferred. This information consists of arithmetic computations, identification labels and display position coordinates. In the case of the computational results, these are taken from the output computation table by the octal-decimal conversion subroutine and inserted directly into the display outline. Data are inserted in the outline in positions which are designated by arguments added to the end of the display outline code. The iterative display process is continued until all table values have been converted and transferred to the display file.

5. INSTRUCTIONS FOR THE OPERATION OF THE COMPUTER DESIGN PROGRAM

The following instructions refer to the operation of a control data corporation model 160 computer with an associated DD65 display system. Figure 26 shows the arrangement of the control keyboard which is called keyboard II. The program is run from cell 100. After activating the track-ball, the track-ball is positioned for the start position of the symbol. The coordinates are entered from the draw button. The track-ball is then moved to the next position and the draw button is again pressed. The coordinates of the new position are entered in the data table and a line appears between the new and the old coordinates. A symbol is drawn in this fashion with its break points recorded in the

data table. If, during this process, a line is drawn which is to be blanked out in the display symbol, the blank button is pressed immediately after drawing the line. Comments may be added or removed at any time by first pressing appropriate button. Typing a carriage return allows normal operations to continue. It is good practice to store the sketched symbol on magnetic tape, since it contains all the information needed for its design. This is done by the write-on magnetic tape button. The block number must be typed in before this action occurs. Design is accomplished by pressing the compute button. After about a second, the circuit diagram for the X plate function generator will appear. Pressing the compute button again will produce the circuit diagram for the Y plate function generator. The circuit diagrams are alternately switched from X and Y by sequentially pressing the compute button. The type of circuit, X or Y, is labeled above the display circuit.

6. COMPUTER DESIGN OF THE LETTER "P"

As an example of the application of computer design, the letter "P" was designed using the Fortran program of Appendix I. The following sentences describe in detail the procedure used to accomplish the design. Firstly, the letter was drawn to scale on paper. The scale was such that the maximum letter size was three hundred units. This was chosen to closely correspond to the size of character which might be drawn on the DD65 display mentioned in section 5 of this chapter. The designed character was to be displayed on a cathode ray oscilloscope on a scale of 0.1 volts per centimeter. The letter size was chosen to be slightly larger than one centimeter. It was decided to sample the current with a ten ohm resistor. From these decisions, the value of k, as defined by

equation 4, is thirty. The value twenty-five was chosen and corresponds to the value of Isens in the Fortran program of Appendix I. The value of Vmax was chosen as twenty volts. This corresponds to a value of two thousand millivolts for IVmax in the Fortran program of Appendix I. The correction for seventy-five ohm power supply internal impedance and ten ohm sampling resistor was taken to be sixty ohms. This correction was discussed in section 1 of Chapter III. With this data, the program was executed and the results obtained are included in Appendix I.

7. CIRCUIT REALIZATION AND DISPLAY OF THE LETTER "P"

Figures 24 and 25 show the circuitry for the function generators required to produce the wave forms necessary for the display of the letter "P". The circuit diagrams corresponding to these which were produced by the DD65 display are shown in figures 27 to 29. The resistance values chosen were those from the Fortran program of Appendix I. A photograph of the symbol generator and ancillary equipment is presented in figure 29. The various parts of the circuitry are labeled in the block diagram of figure 30 and listed in Table I. Figures 31 to 35 show various wave forms of the symbol generator circuit. It should be observed that the sweep input of figure 31 was not of perfectly uniform intensity which caused a slight bit of intensity variation in the character. This was due to the necessity of intensity modulating the oscilloscope to remove the distortions and the idle part of the sweep of figure 30. The imperfect intensity modulation is apparent from figures 31 and 32.

TABLE I

TABLE OF EQUIPMENT USED WITH SYMBOL FUNCTION GENERATORS

TO PRODUCE SAMPLE CHARACTER "P"

QUANTITY	EQUIPMENT	MANUFACTURER	REMARKS
1	Function Generator	Wavetek, Model 112	Sawtooth Function used
2	Power Amplifier	Hewlett Packard, Model 467A	0-20 Volts, 20db gain, 0-1MHz
1	Unit Pulse Generator	General Radio, Type 1217-B	
1	Oscilloscope	Hewlett Packard, Model 20B	
2	Power Supply	Hewlett Packard, Model 721A	0-30 Volts, 0-225 milli-amperes
1	Impedance Bridge	General Radio, Type 650A	

CHAPTER IV

DISCUSSION OF RESULTS AND CONCLUSIONS

1. DISCUSSION OF THE RESULTS

The results of the example design problem were partially discussed in section 7 of Chapter III. It is seen from figure 35 that this design procedure yields a very acceptable character. It is noted that the character display time is two hundred microseconds. While this is comparatively slow, tests were made at twenty microseconds that were encouraging. These tests are not included since sufficient error existed in the input sweep voltage wave form at a twenty microsecond sweep time to invalidate the results.

2. CONCLUSIONS

The procedure developed for symbol design proved to be easy to follow and gave a solution which was suitable for output to a plotter. More flexibility could have been obtained at the expense of larger core storage. The economy of core storage utilized is illustrated in Table II. Programing effort would have been considerably reduced if a separately stored library of service subroutines were available. In this regard, it is felt that the output display also could have been presented more easily if there were library stores of code to be used for display of circuit components. Using this method, output display could be built up with little programing effort using the canned circuit component code. In addition to the above remarks, it is also felt that the existence of a supervisory control subroutine, controlling arithmetic accuracy, also would have reduced the programing load. Such a subroutine would not necessarily have to use floating-paint techniques, but would look after

TABLE II

ALLOCATION OF CORE STORAGE FOR COMPUTER

PROGRAM OF APPENDIX II

SUBROUTINE	CORE STORAGE	SUBROUTINE	CORE STORAGE
Console	100-116	Disply	2000-2240
Trkbal	117-153	r/wr	2450-2515
tb	155-172	Read/Write	2517-2551
drw	173-263	Blank	2555-2575
Line 1	257-507	Octdec	2600-2761
Print	516-710	S Root	2770-3073
Dis	750-1034	Main Outline	3100-3145
Enter	1050-1145	Pos Outline	3200-3257
Remove	1150-1224	Neg Outline	3300-3357
Comput	1250-1740	Tables & Files	3600-3777
Buffer) Temporary) s/r Jump) Index)	0-77		

the maintenance of accuracy and ensure that large arithmetic errors, such as overflow occurring in multiplication operations, did not occur. Finally, it is regretted that the software was not available to output the solution in hard copy form. In spite of the above shortcomings of available computer service subroutines and peripheral equipment, the procedure demonstrated the ease and flexibility with which this type of design may be accomplished.

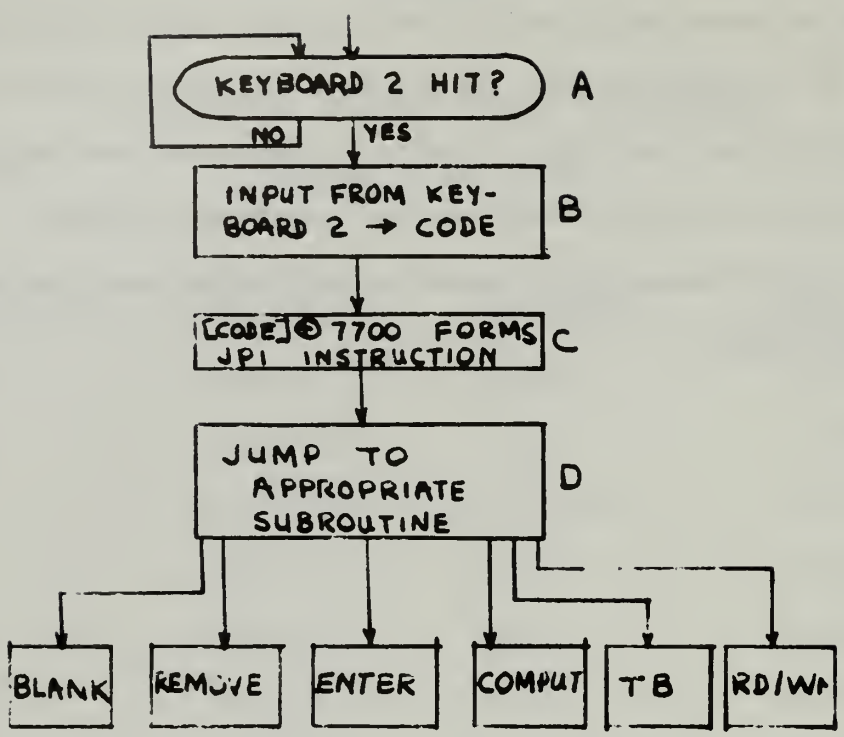

FIGURE 10

CONSOLE SUBROUTINE PROVIDES CONSOLE CONTROL

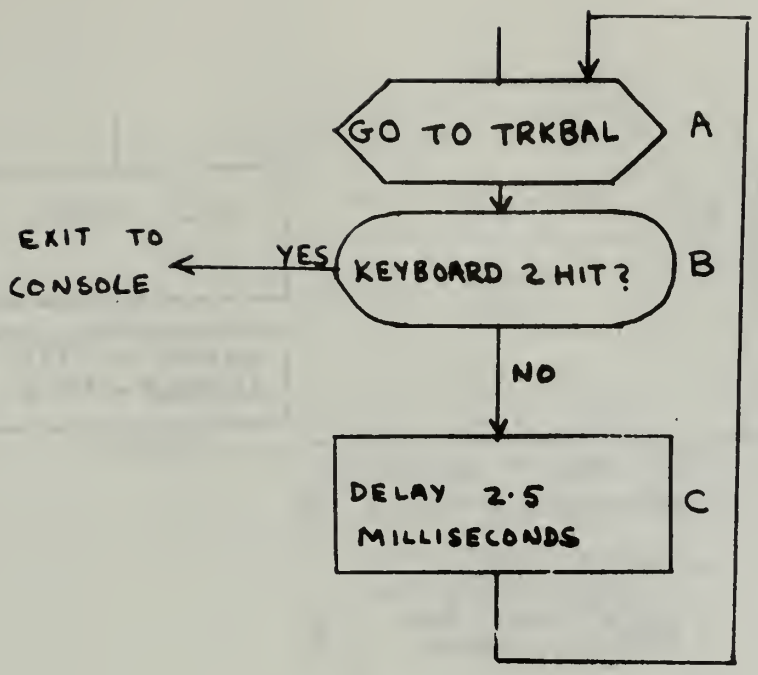

FIGURE 11

TB SUBROUTINE PROVIDES THE ORGANIZATION TO DRIVE THE TRACK-BALL SUBROUTINE

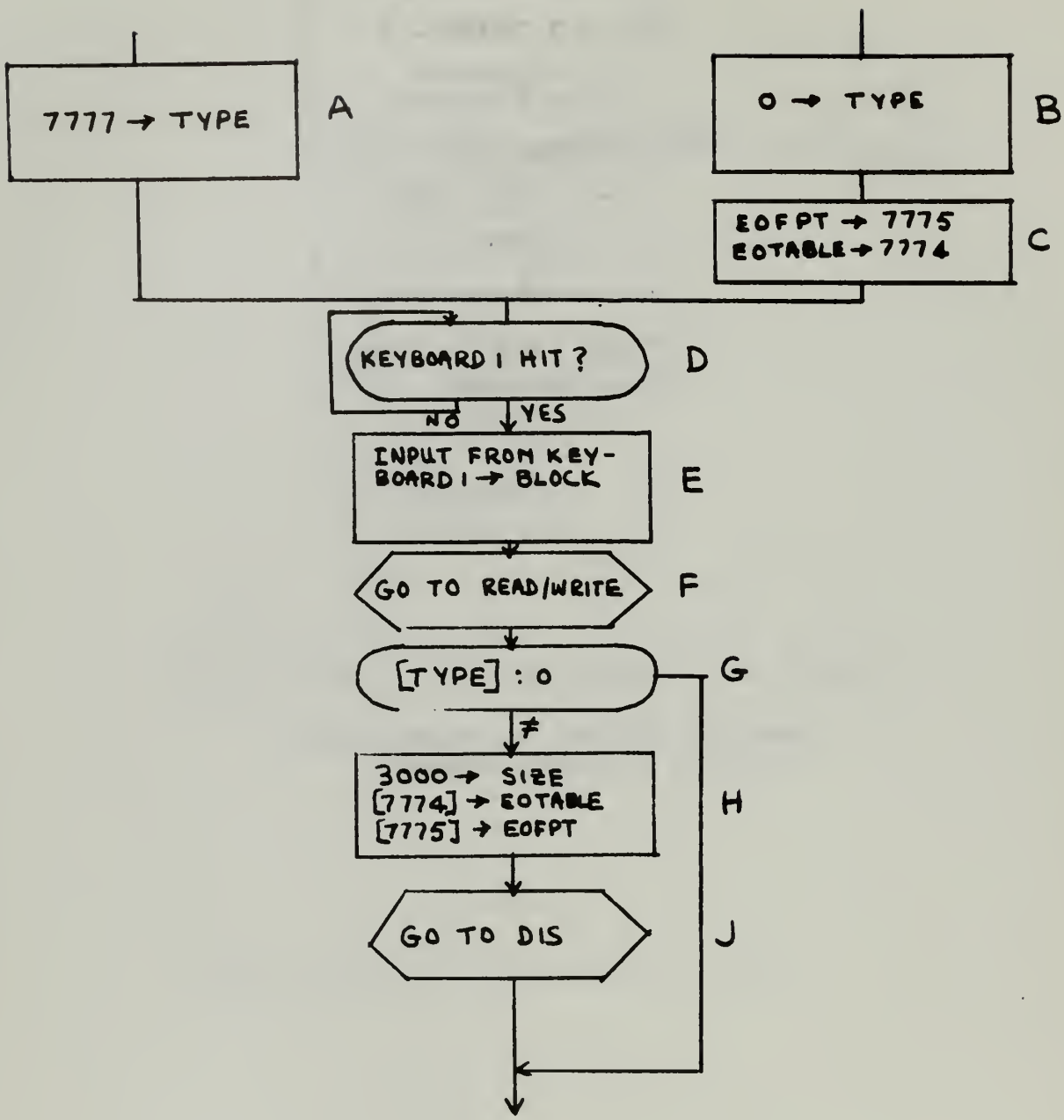

FIGURE 12

RD/WR SUBROUTINE CONTROLS THE USE OF THE MAGNETIC TAPE UNIT

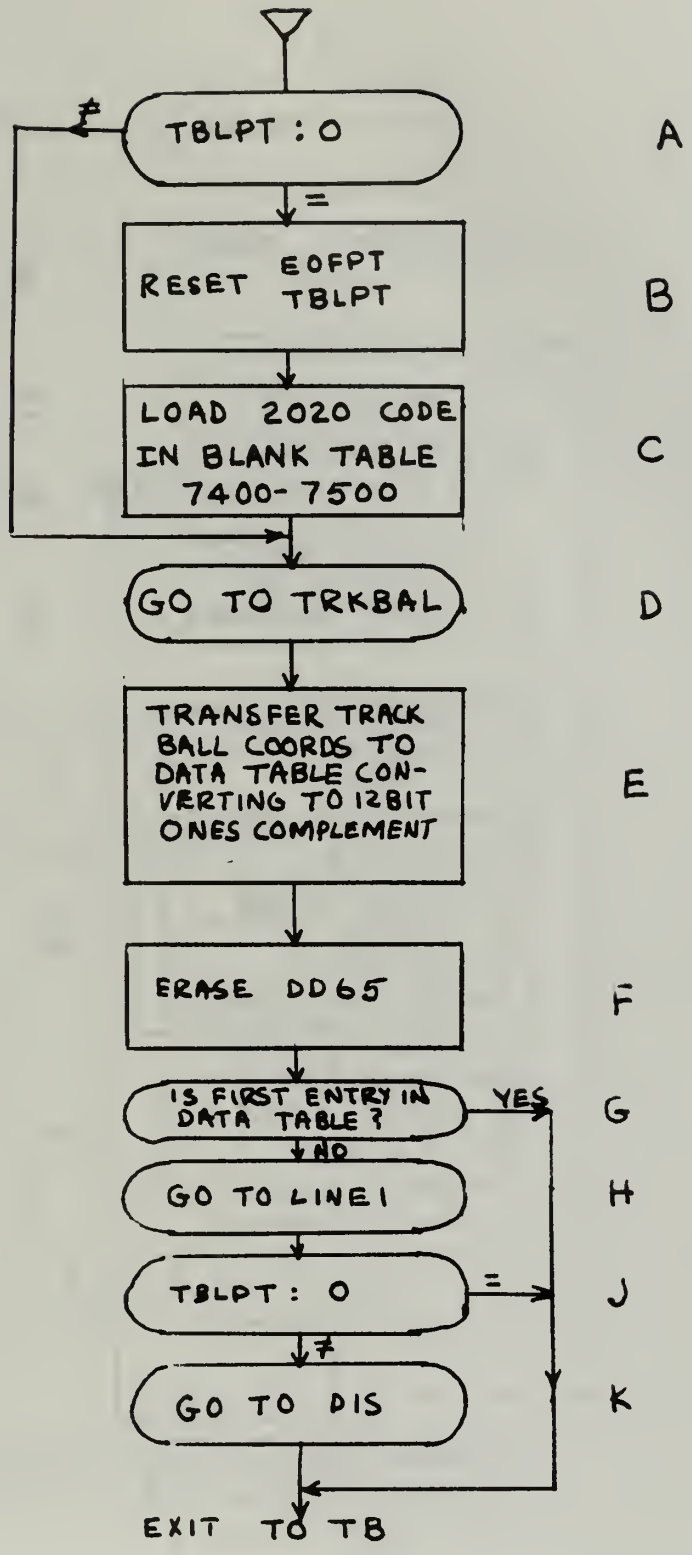

FIGURE 13

DRW SUBROUTINE CAUSES DATA TO BE ENTERED INTO THE DATA TABLES AND ACTS AS A DRIVER FOR THE INPUT DATA DISPLAY DRIVER WHICH IS CALLED LINE 1.

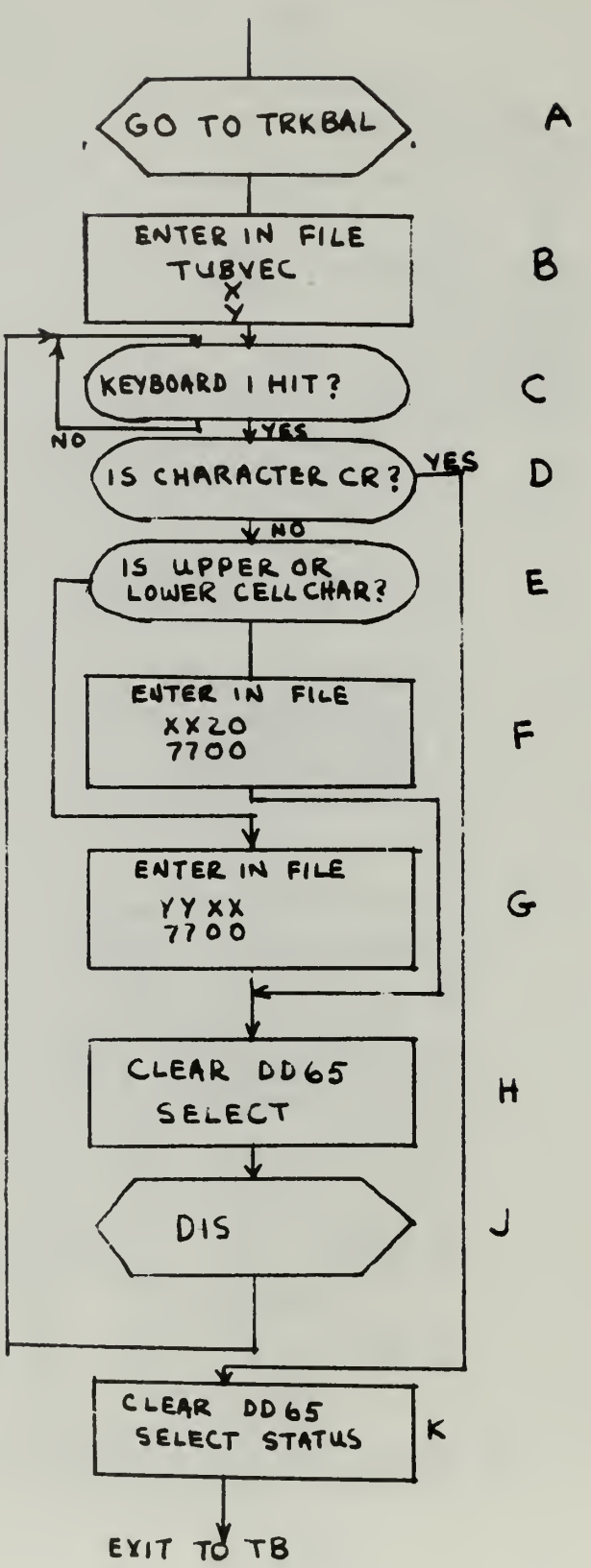

FIGURE 14

ENTER SUBROUTINE ADDS COMMENTS TO
THE DISPLAY FILE

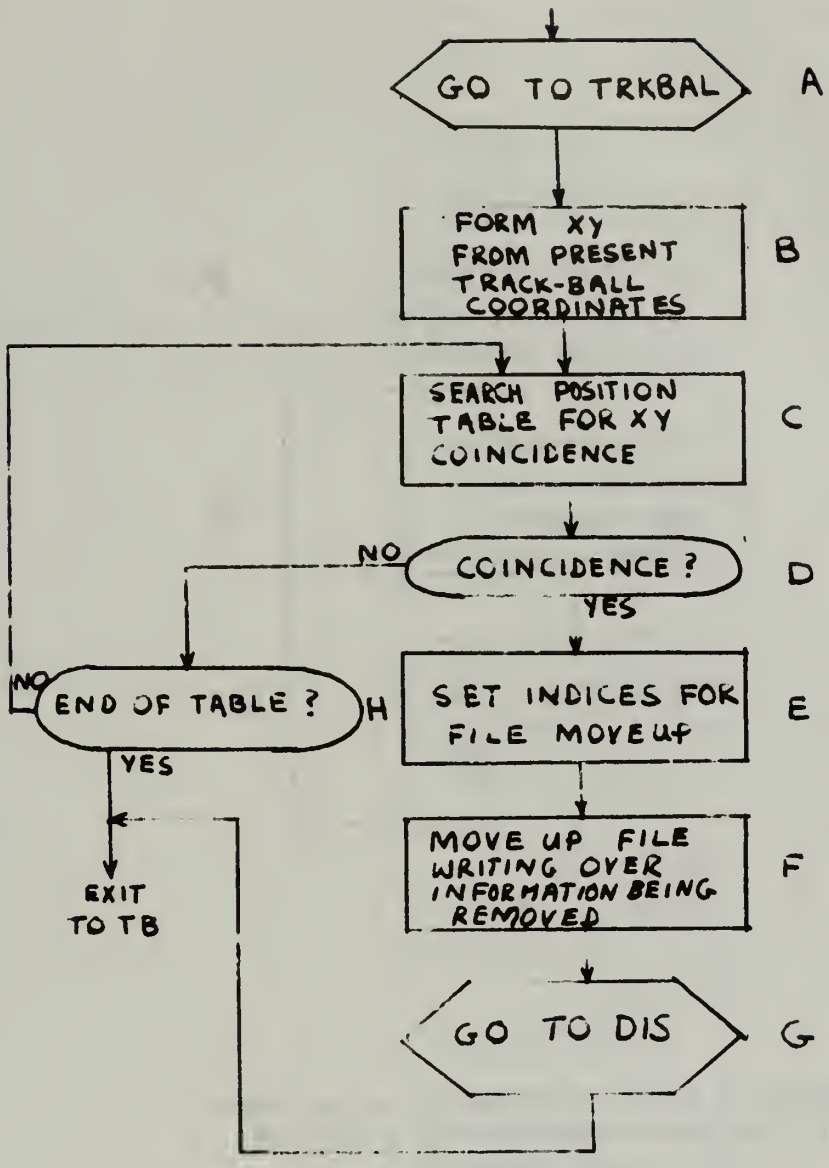

FIGURE 15

REMOVE SUBROUTINE REMOVES PREVIOUSLY ENTERED INFORMATION FROM THE DISPLAY FILE

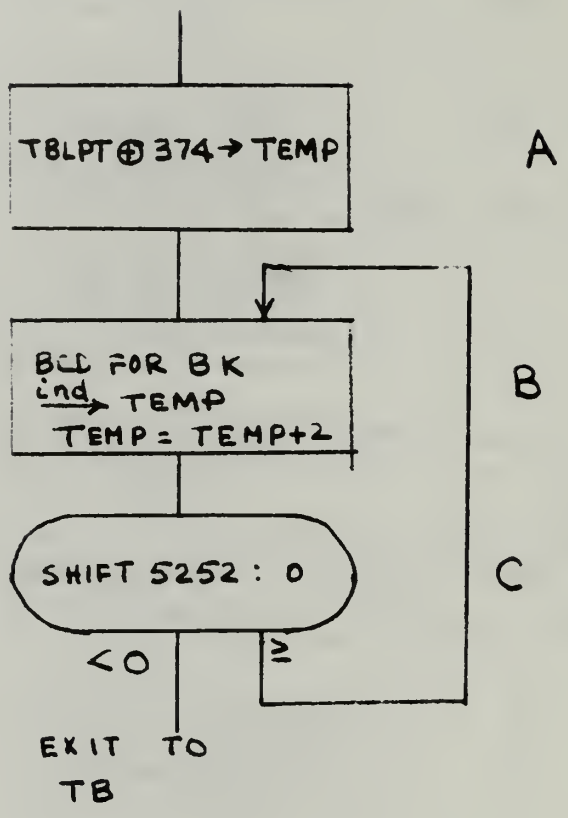

FIGURE 16

BLANK SUBROUTINE ADDS THE LETTER BK IN THE BLANK FILE IF BLANKING IS REQUIRED FOR A PORTION OF THE SYMBOL GENERATED

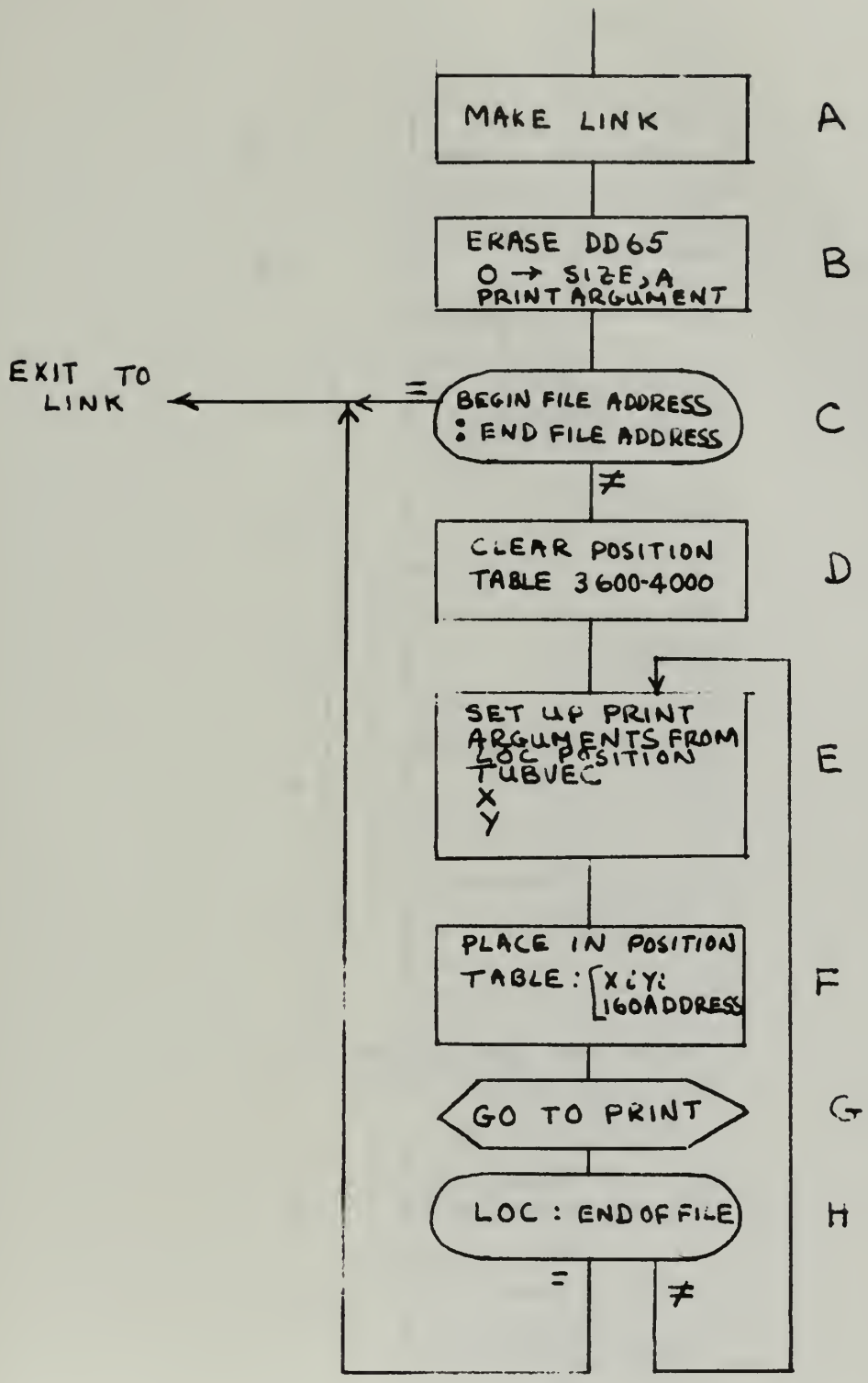

FIGURE 17

DIS SUBROUTINE DISPLAYS THE FILE

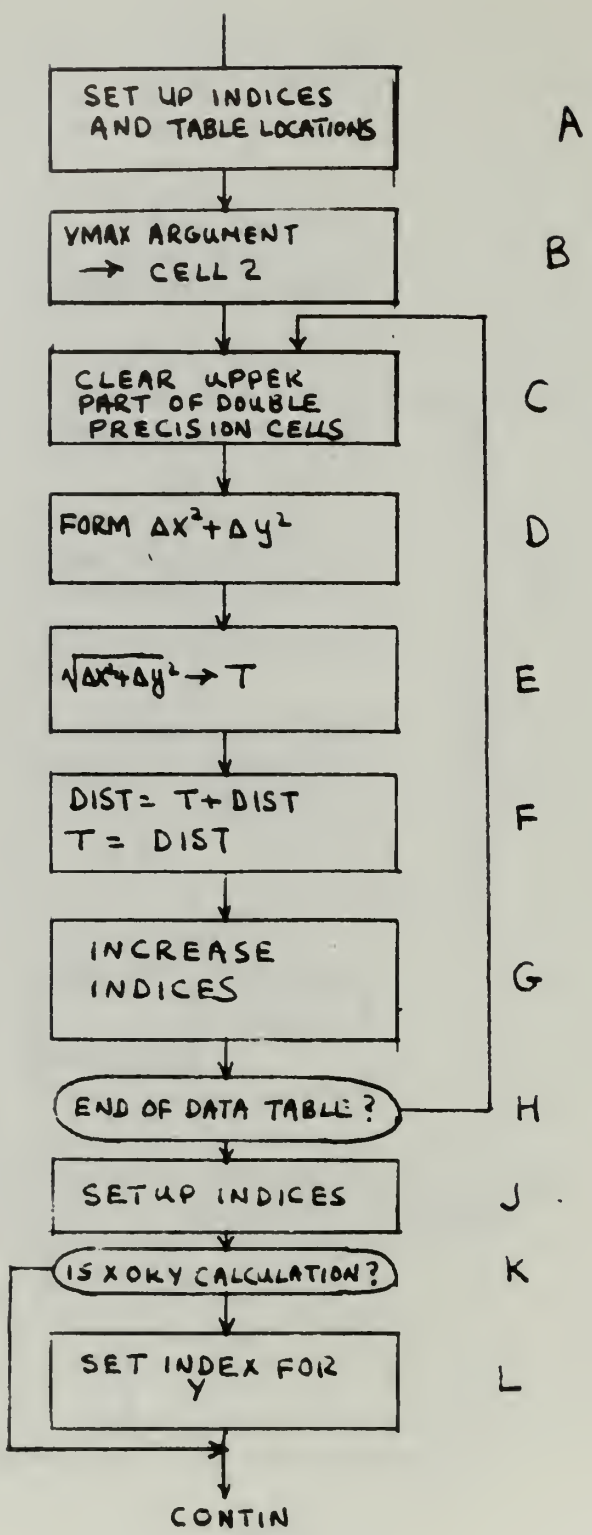

FIGURE 18

COMPUT SUBROUTINE IS THE MAIN PROGRAM WHICH CONTAINS THE ALGORITHM FOR DESIGN OF THE SYMBOL GENERATORS

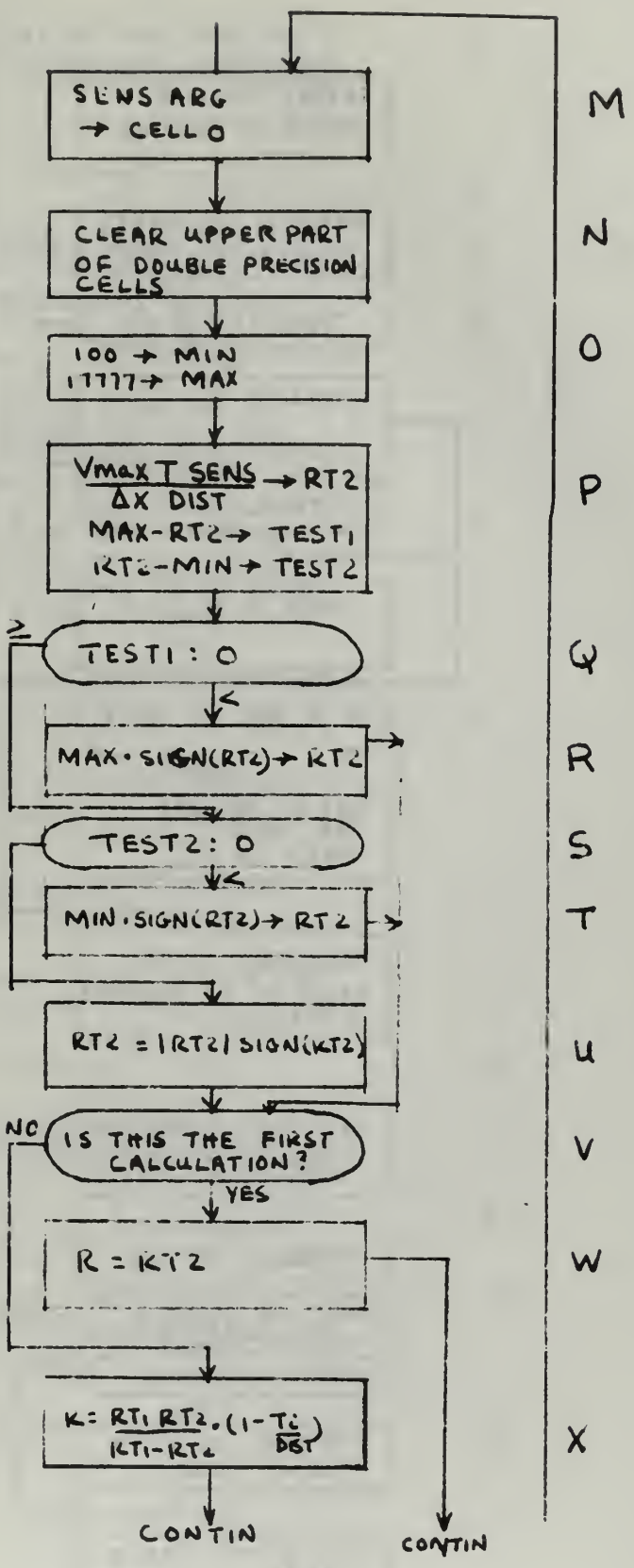

FIGURE 18 CONTINUED

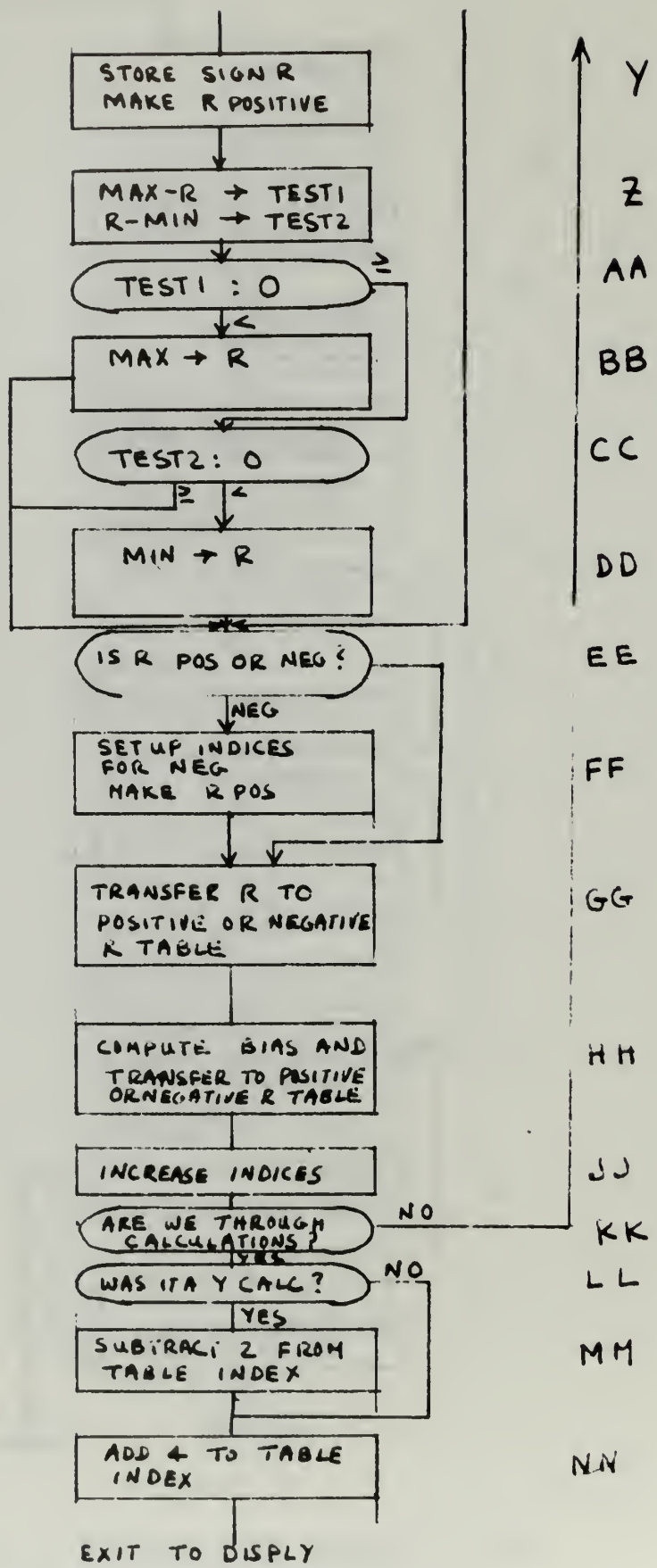

FIGURE 18 CONTINUED

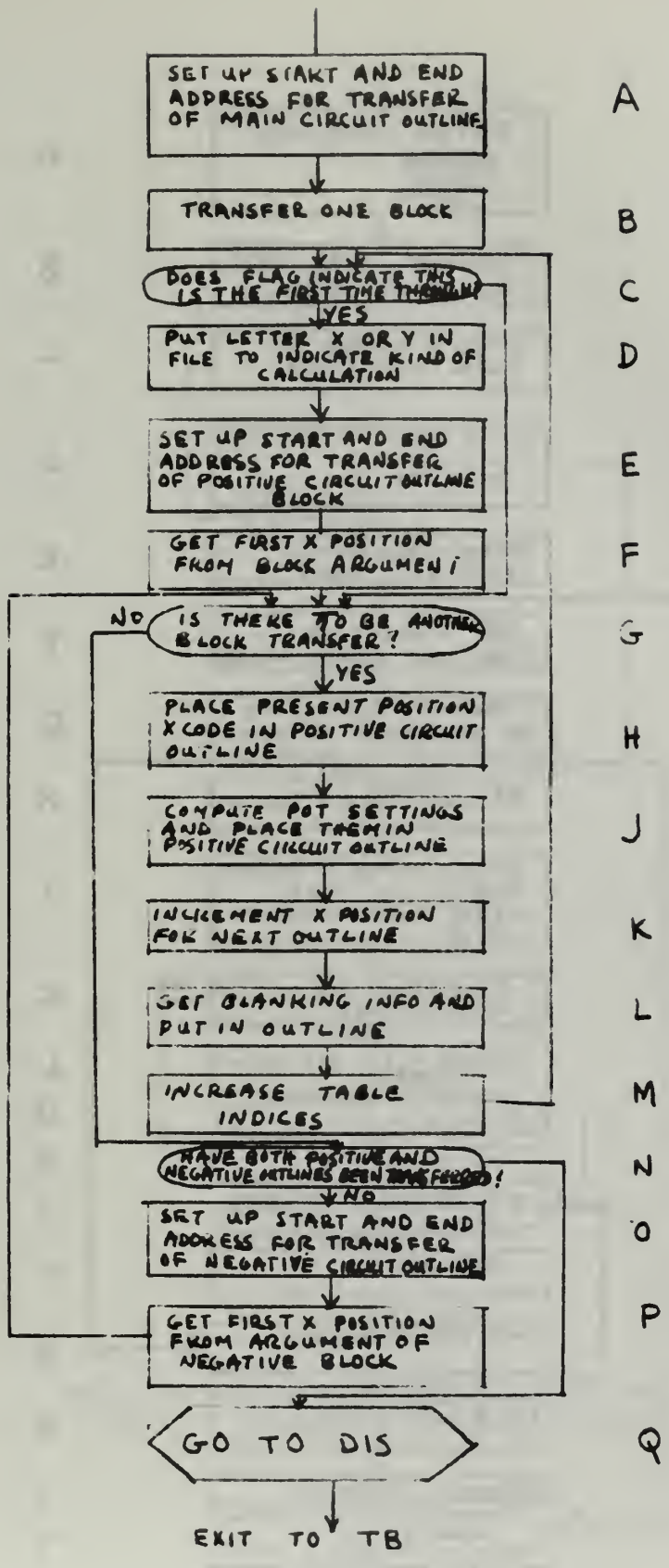

FIGURE 19

DISPLY SUBROUTINE CONVERTS RESISTANCE AND DIODE BIAS VALUES TO POTENTIOMETER SETTINGS AND COORDINATES THEIR DISPLAY WITH THE CIRCUIT DIAGRAM

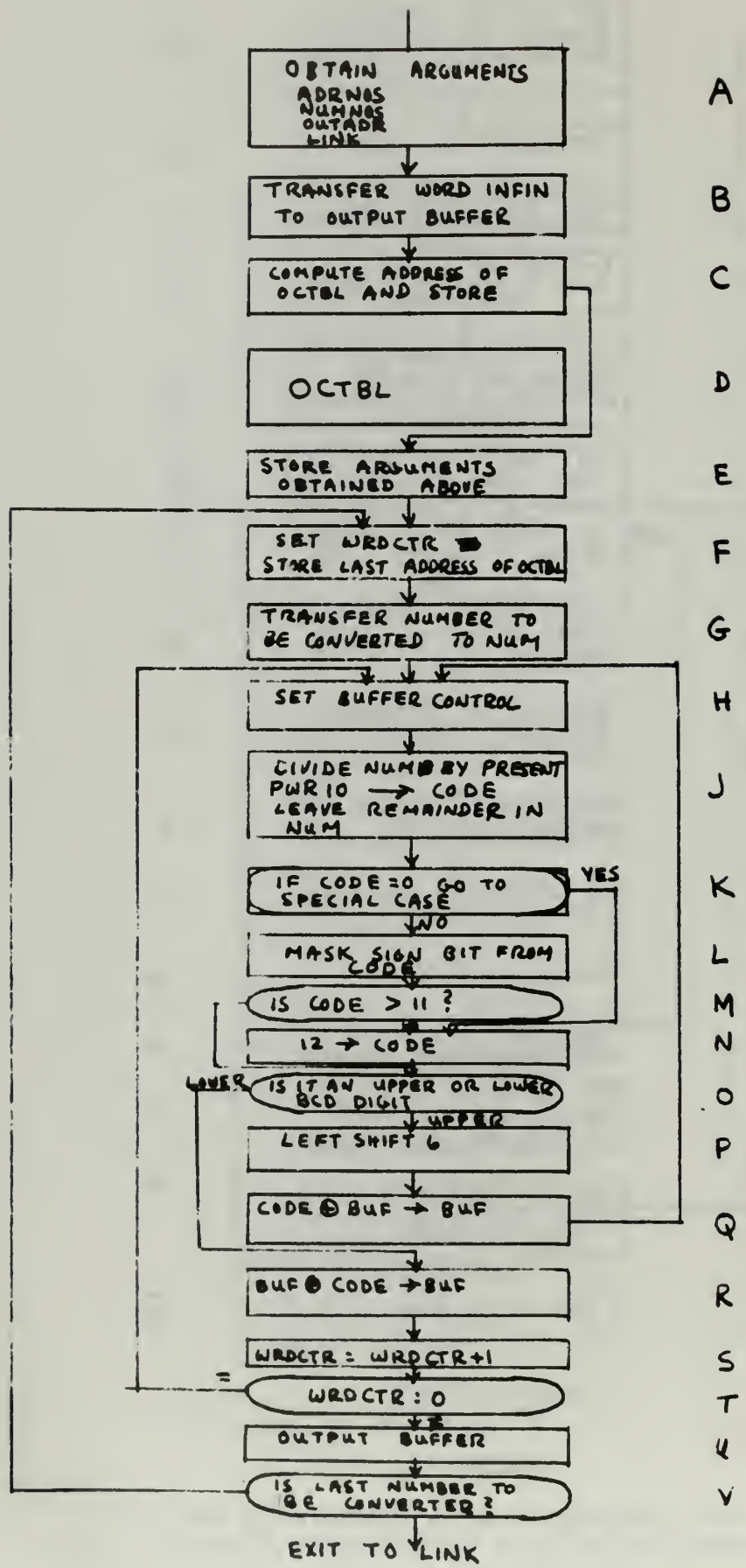

FIGURE 20

OCTDEC SUBROUTINE CONVERTS 22 OCTAL NUMBERS
TO 36 BITS OF BCD CODE

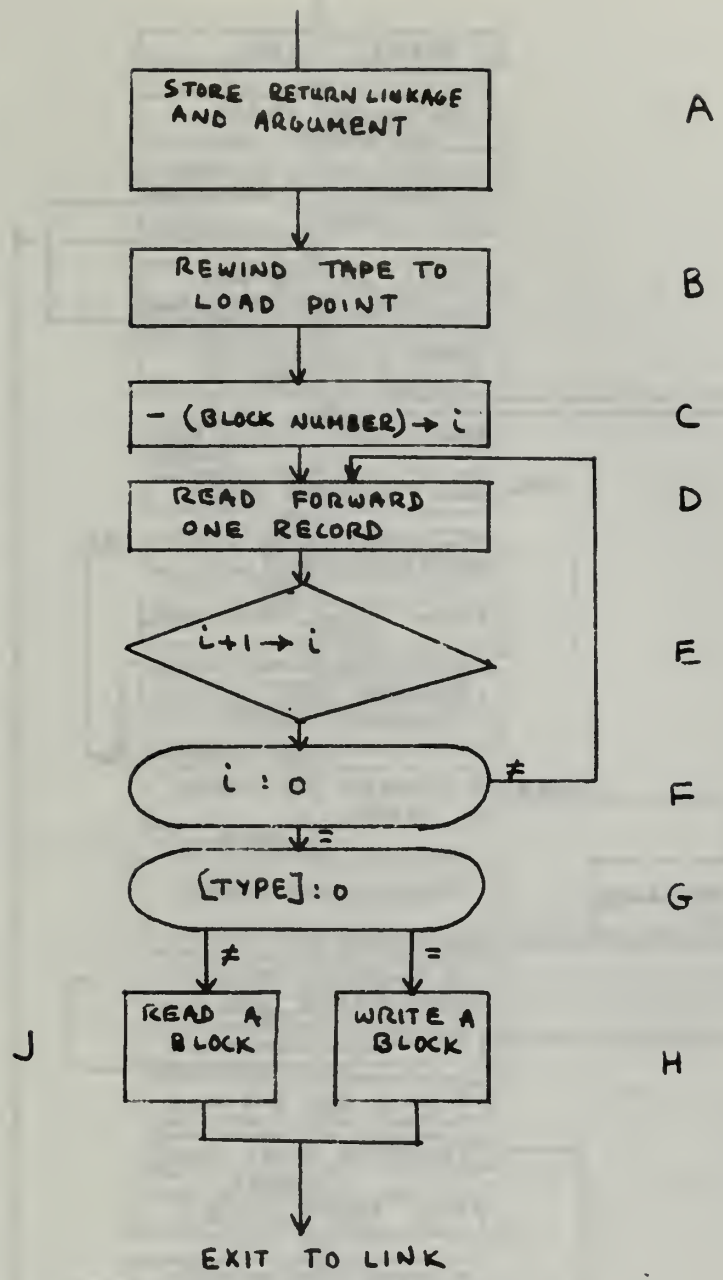

FIGURE 21

READ/WRITE SUBROUTINE READS OR WRITES ONE BLOCK OF INFORMATION ON MAGNETIC TAPE

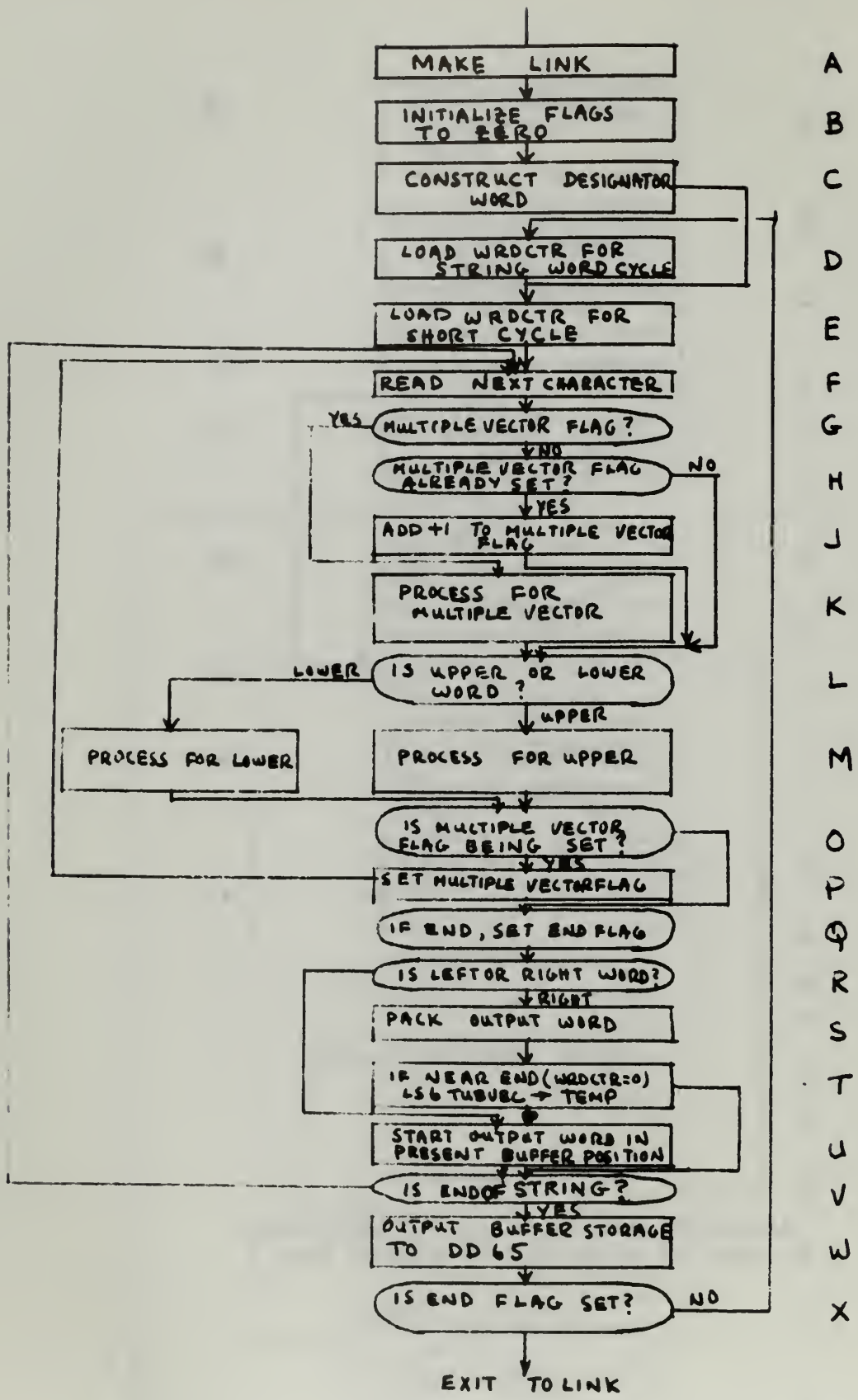

FIGURE 22

PRINT SUBROUTINE TAKES A STRING OF VECTOR OR CHARACTER CODE AND TRANSFERS IT TO THE DD 65 DISPLAY

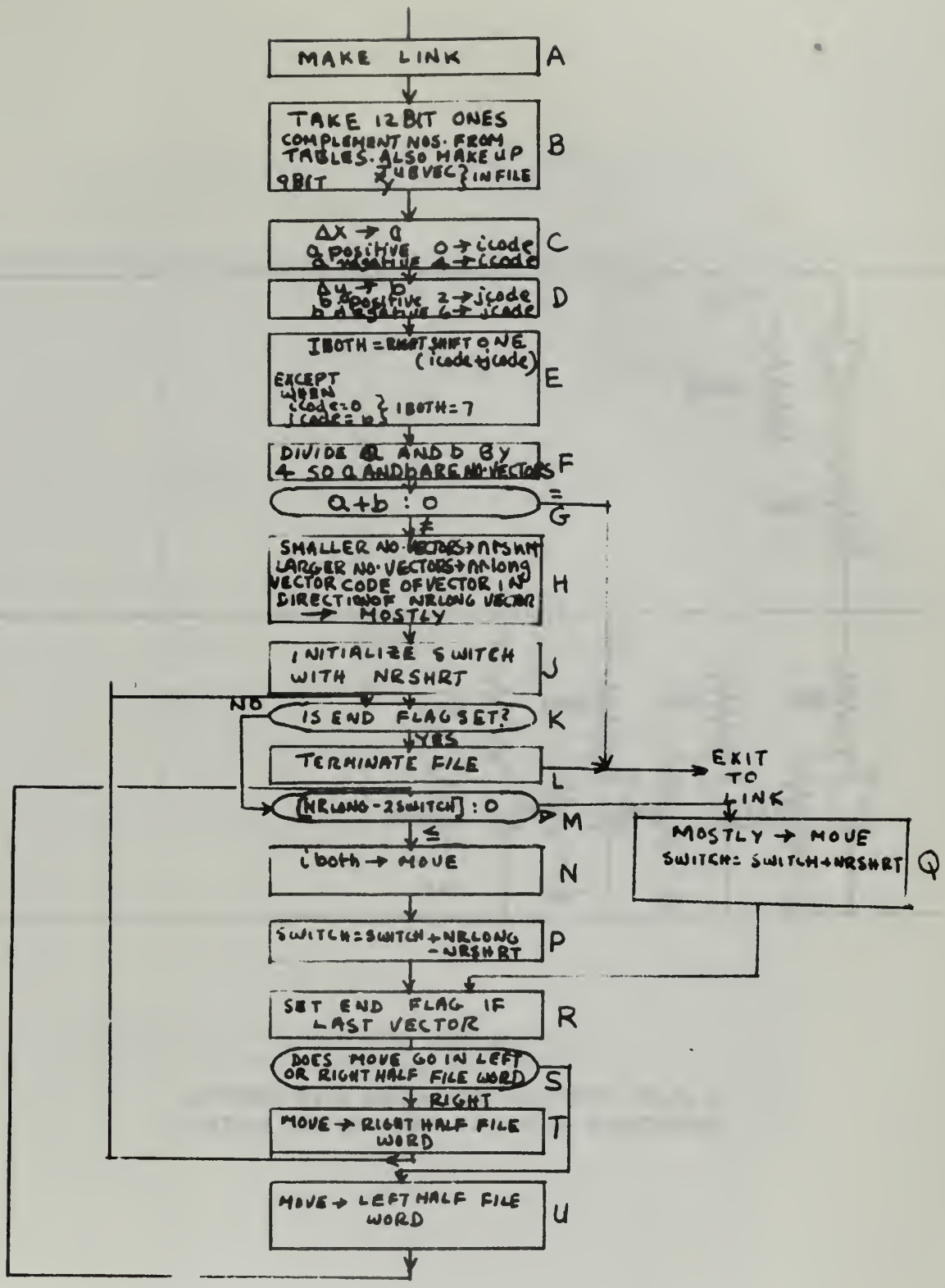

FIGURE 23

LINE 1 SUBROUTINE TAKES X AND Y COORDINATES FROM A TABLE AND SETS UP CODE FOR SUBROUTINE PRINT IN A FILE FOR DISPLAY OF A LINE OF VECTORS BETWEEN THE COORDINATE PAINTS

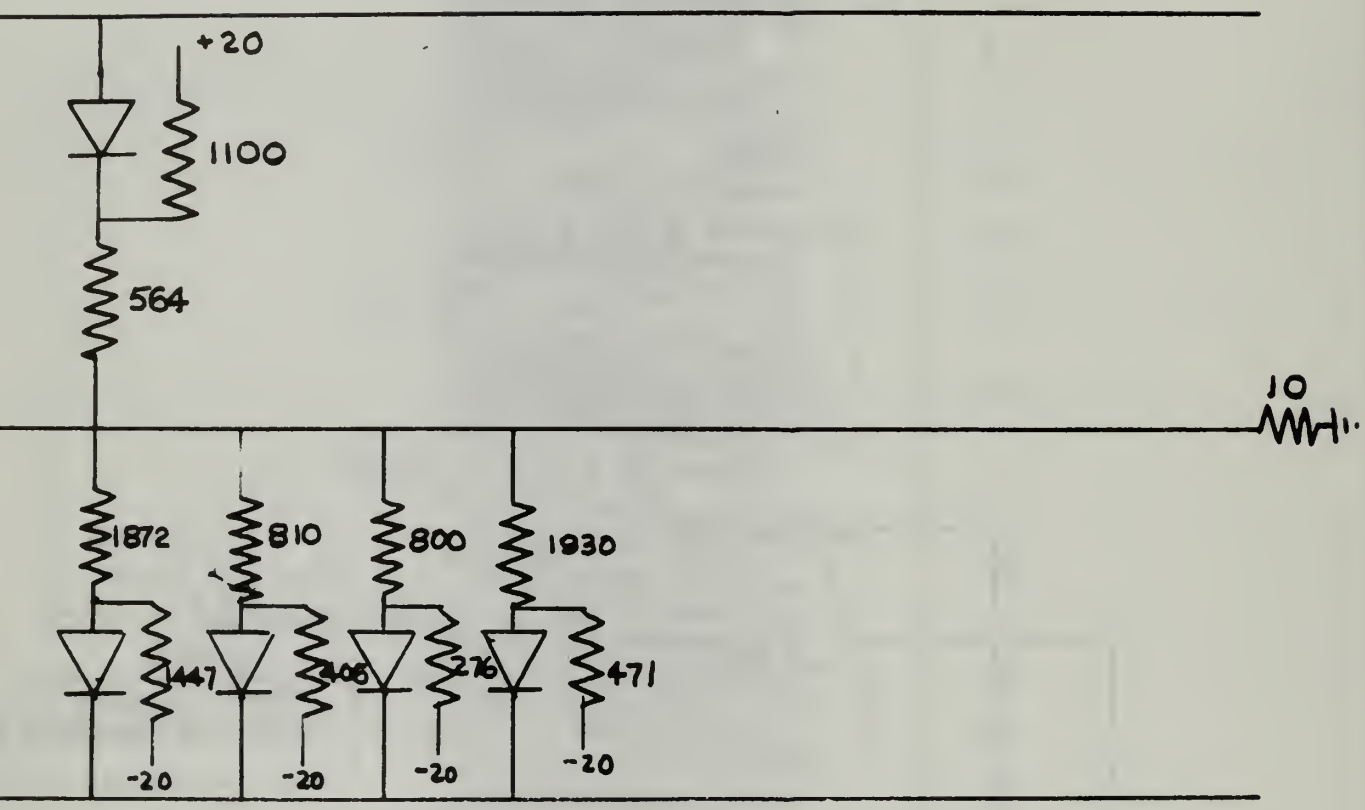

FIGURE 24

X PLATE FUNCTION GENERATORS WITH COMPUTED
RESISTANCE VALUES IN OHMS FOR THE LETTER P

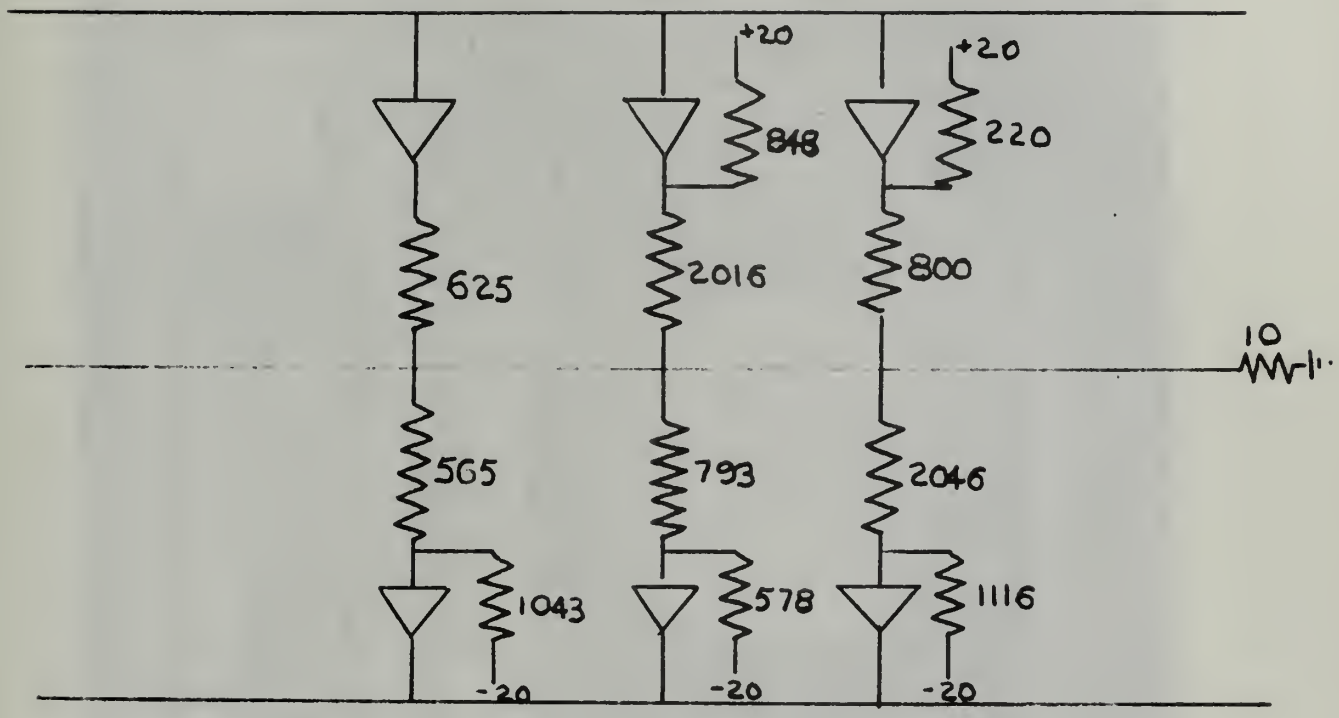

FIGURE 25

Y PLATE FUNCTION GENERATORS WITH COMPUTED RESISTANCE VALUES IN OHMS FOR THE LETTER P

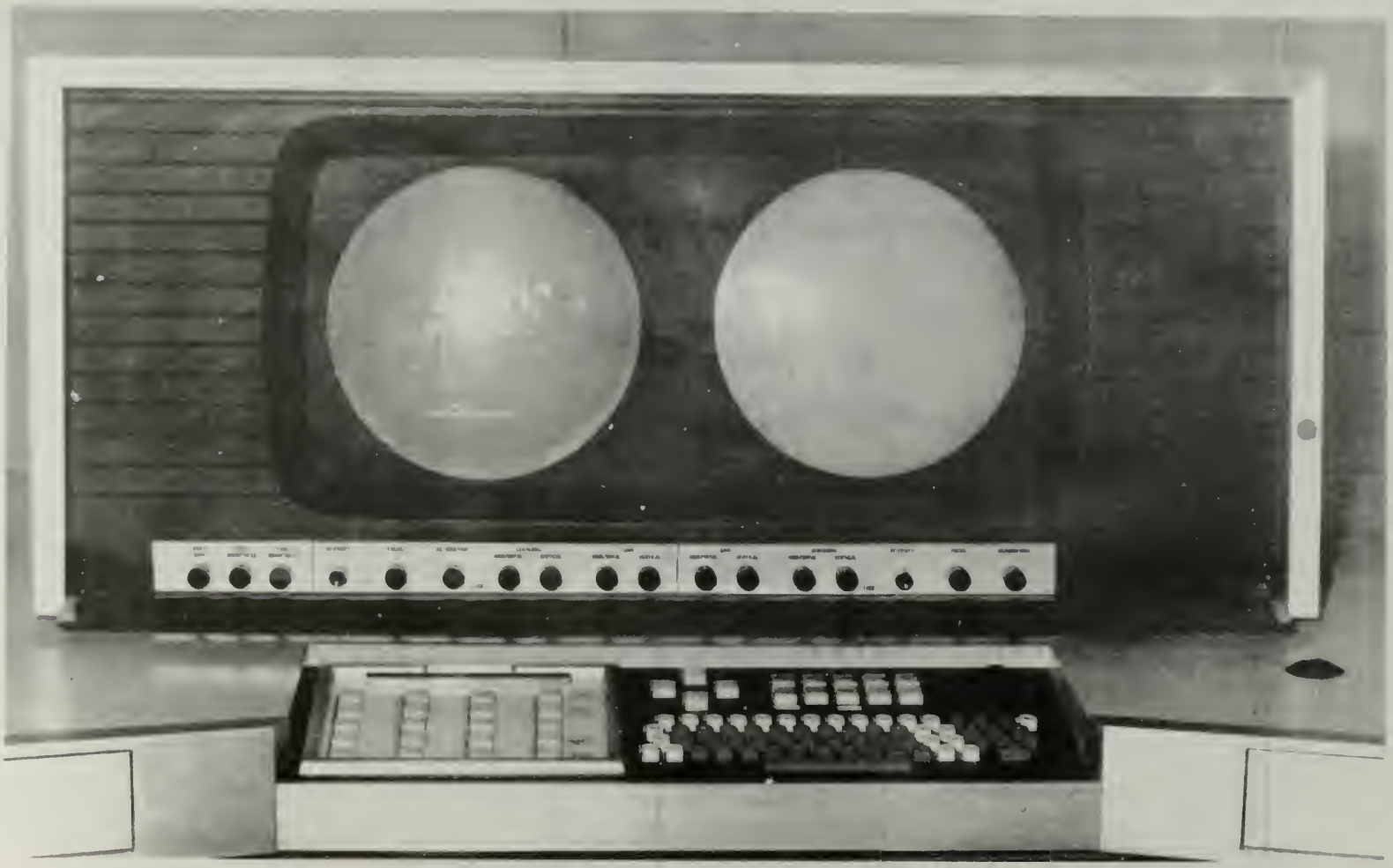

FIGURE 26

PHOTOGRAPH OF DEC DISPLAY CONSOLE

FIGURE 27

PHOTOGRAPH OF DD65 DISPLAY OF INFORMATION
INPUT TO THE COMPUTER FOR THE DESIGN OF THE
LETTER "P"

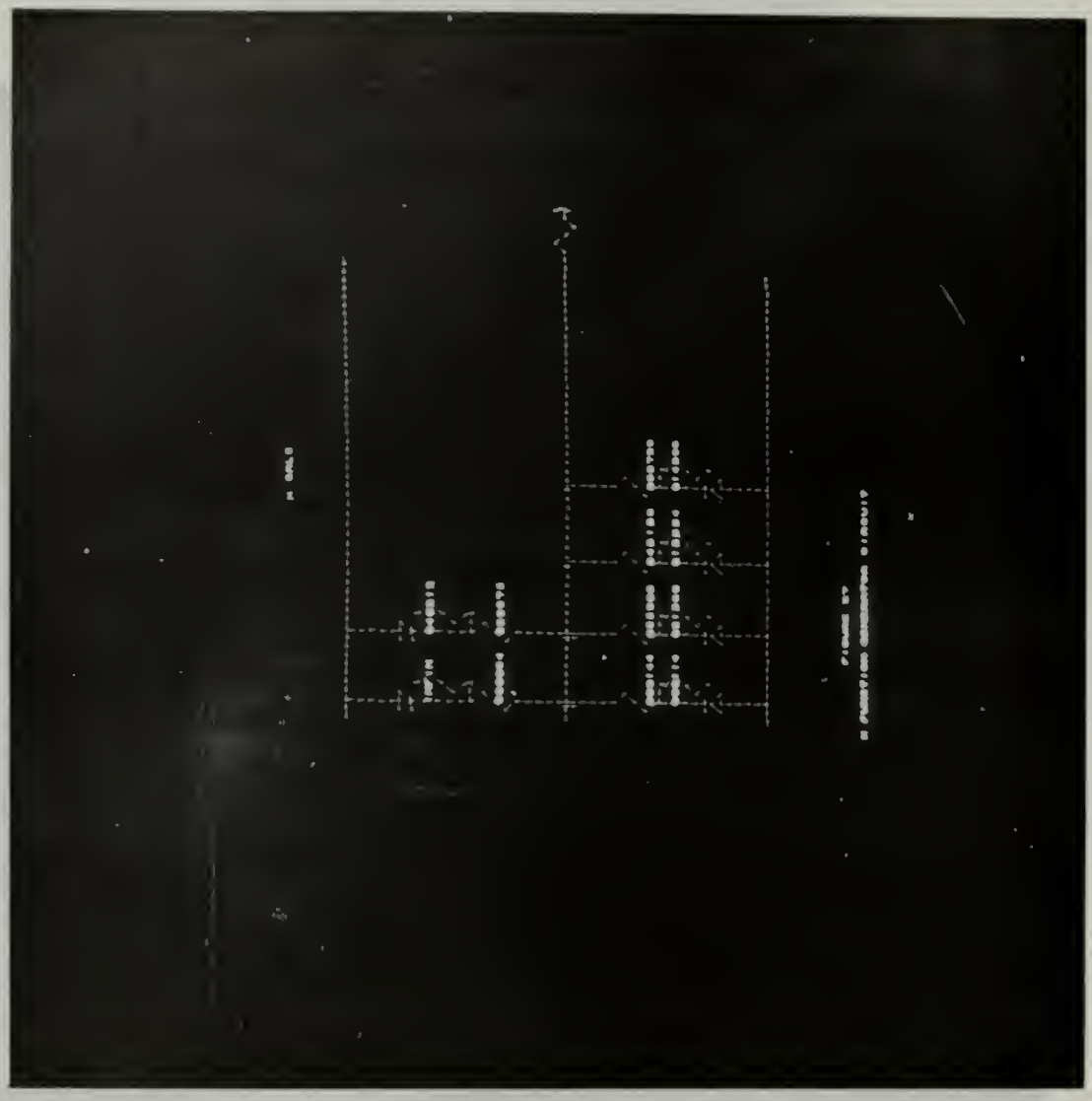

FIGURE 28

PHOTOGRAPH OF THE COMPUTER DESIGNED CIRCUITRY
FOR THE X SYMBOL GENERATOR

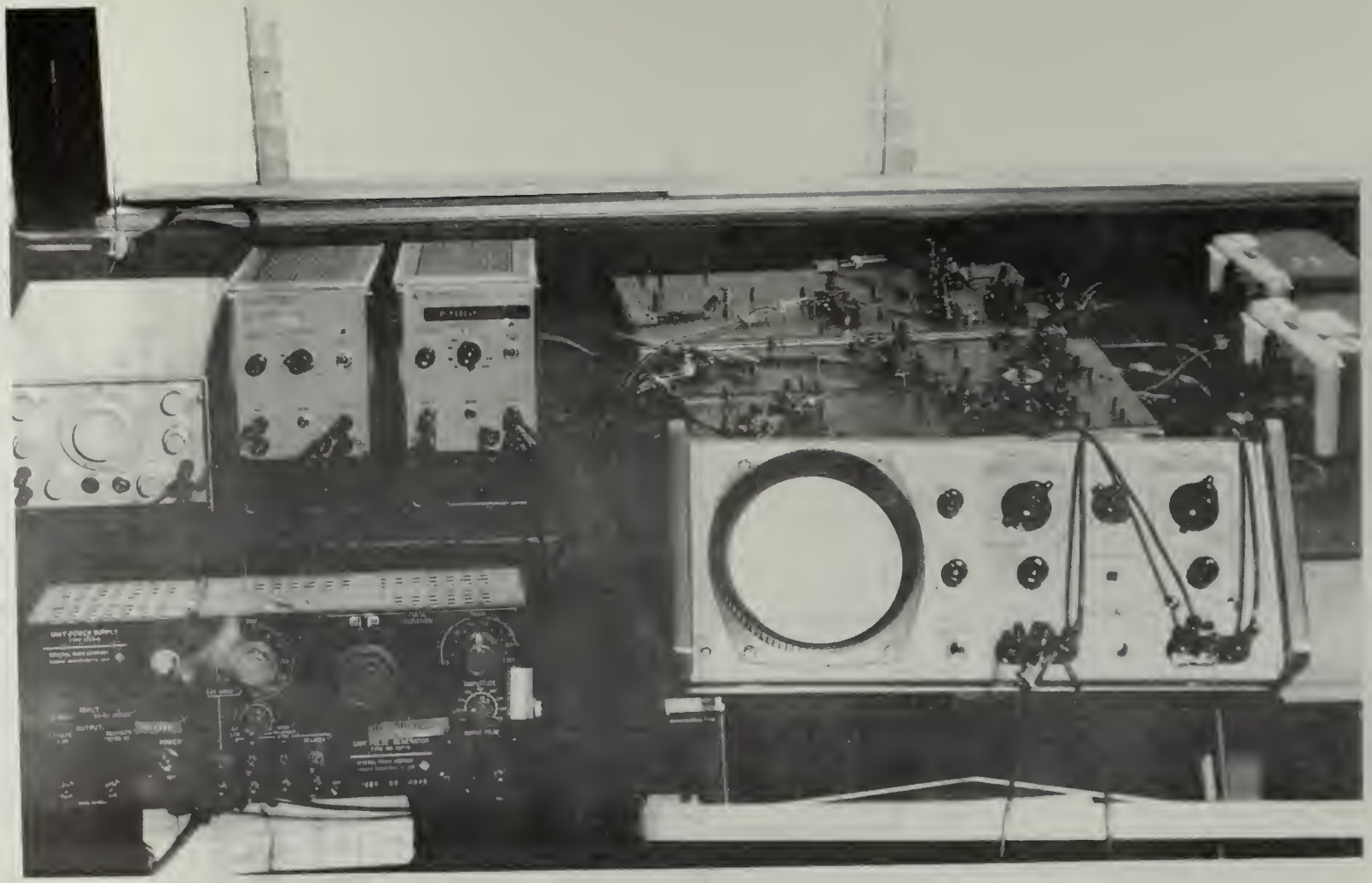

FIGURE 29 PHOTOGRAPH OF THE COMPUTER DESIGNED CIRCUITRY PLUS ANCILLARY EQUIPMENT ASSEMBLED TO PRODUCE THE LETTER "P"

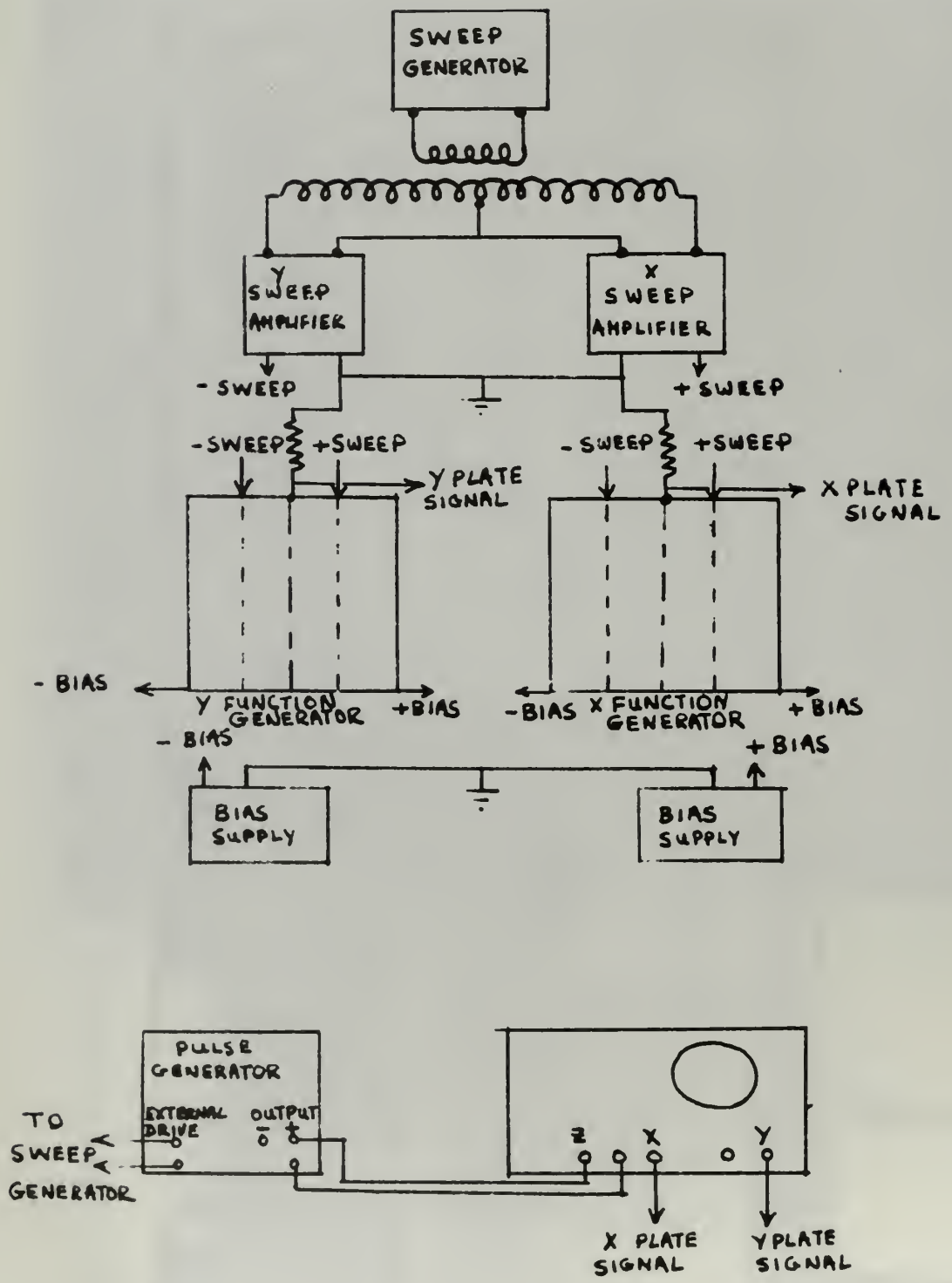

FIGURE 30

BLOCK DIAGRAM OF CIRCUITRY AND EQUIPMENT
USED TO DISPLAY THE LETTER "P"

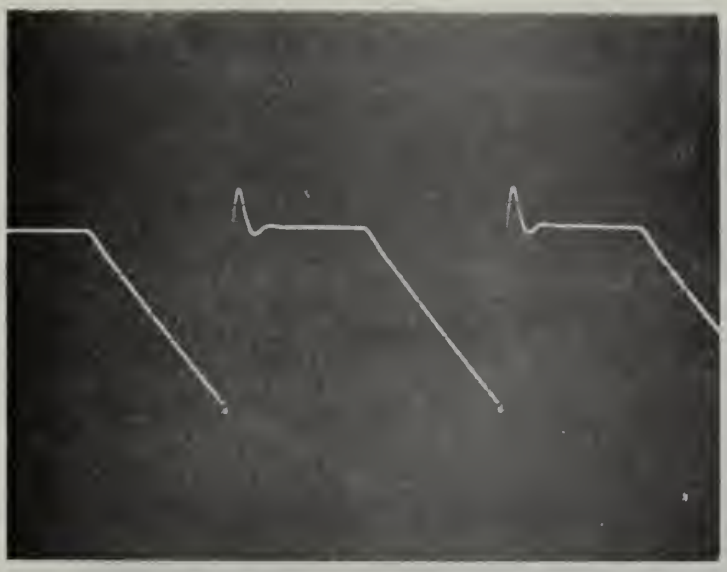

FIGURE 31

FUNCTION GENERATOR NEGATIVE SWEEP SIGNAL. HORIZONTAL SCALE 100 MICRO SECONDS PER CENTIMETER. VERTICAL SCALE 10 VOLTS PER CENTIMETER

FIGURE 32

FUNCTION GENERATOR NEGATIVE SWEEP SIGNAL WITH INTENSITY MODULATION

FIGURE 33

X FUNCTION GENERATOR OUTPUT SIGNAL
HORIZONTAL SCALE 100 MICRO SECONDS PER CM.
VERTICAL SCALE 100 MILLIVOLTS PER CM.

FIGURE 34

Y FUNCTION GENERATOR OUTPUT SIGNAL
HORIZONTAL SCALE IS 100 MICRO SECONDS PER CM.
VERTICAL SCALE 100 MILLIVOLTS PER CM.

FIGURE 35

CHARACTER "P" GENERATED IN ABOUT 200 MICRO SECONDS WITH BOTH HORIZONTAL AND VERTICAL SCALE OF 100 MILLIVOLTS.

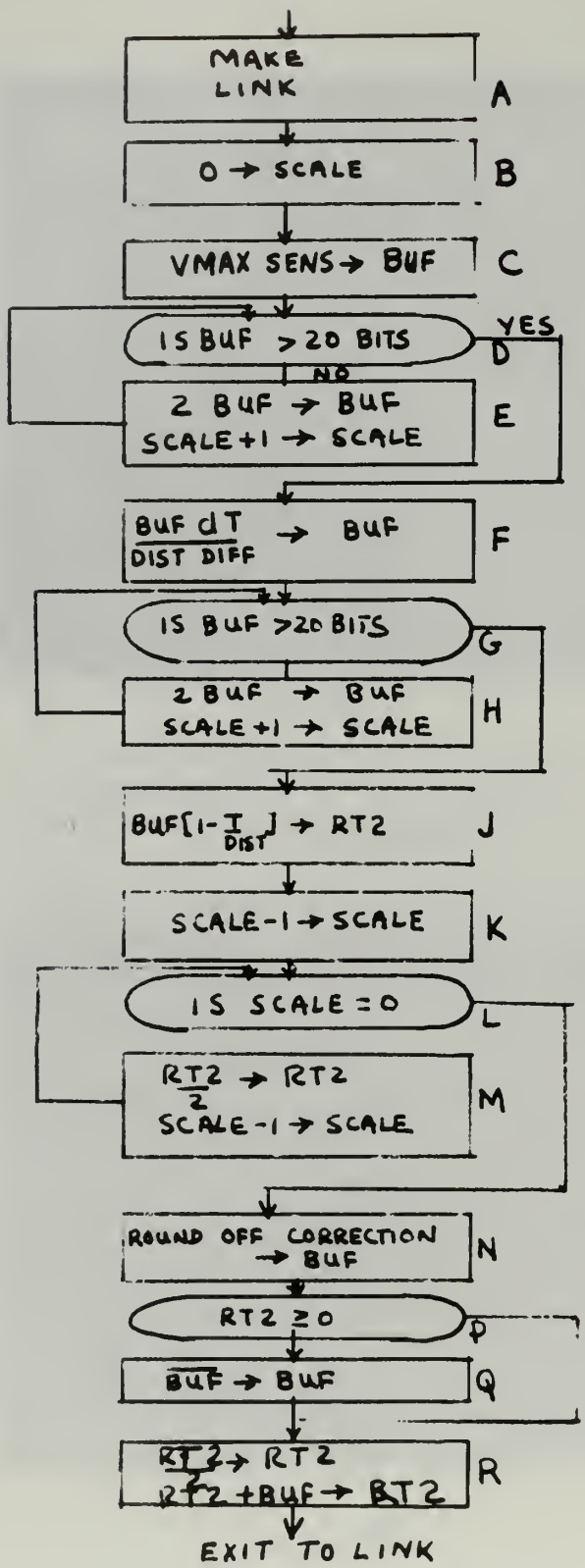

FIGURE 36

SUBROUTINE FIXDPT CARRIES OUT A PARTICULAR SEQUENCE OF FIXED POINT ARITHMETIC COMPUTATIONS

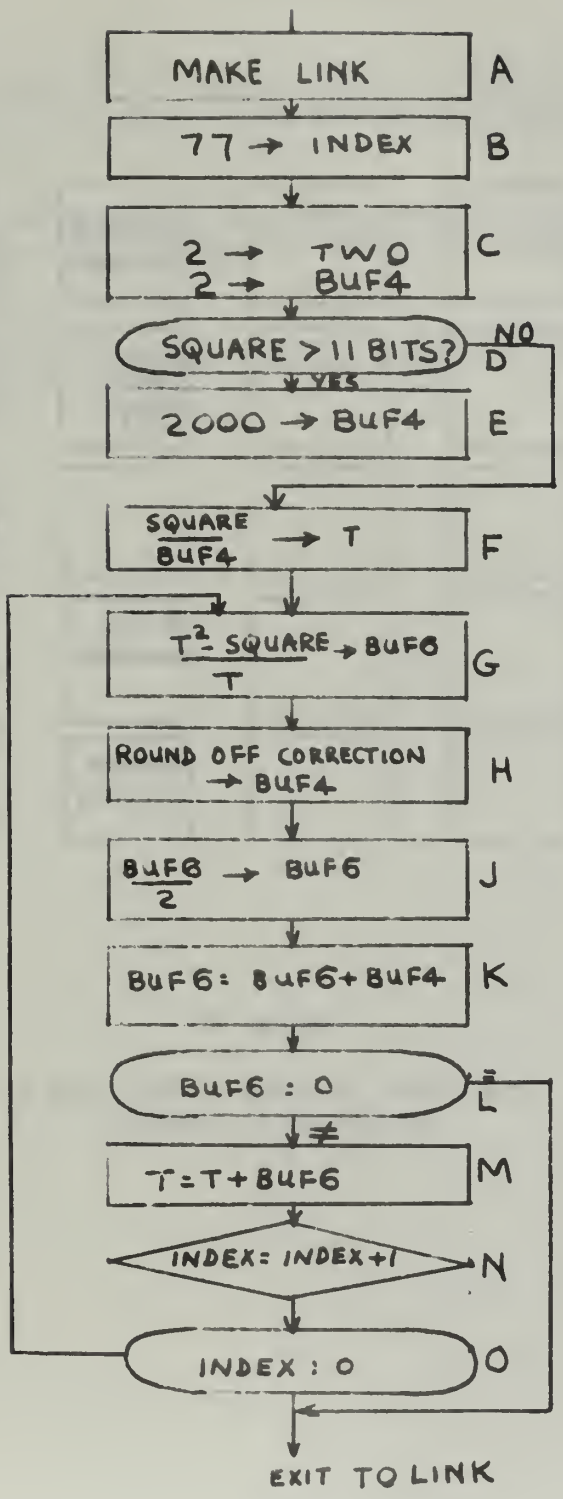

FIGURE 37

SUBROUTINE SROOT COMPUTES THE SQUARE ROOT
OF A FIXED POINT VARIABLE SQUARE

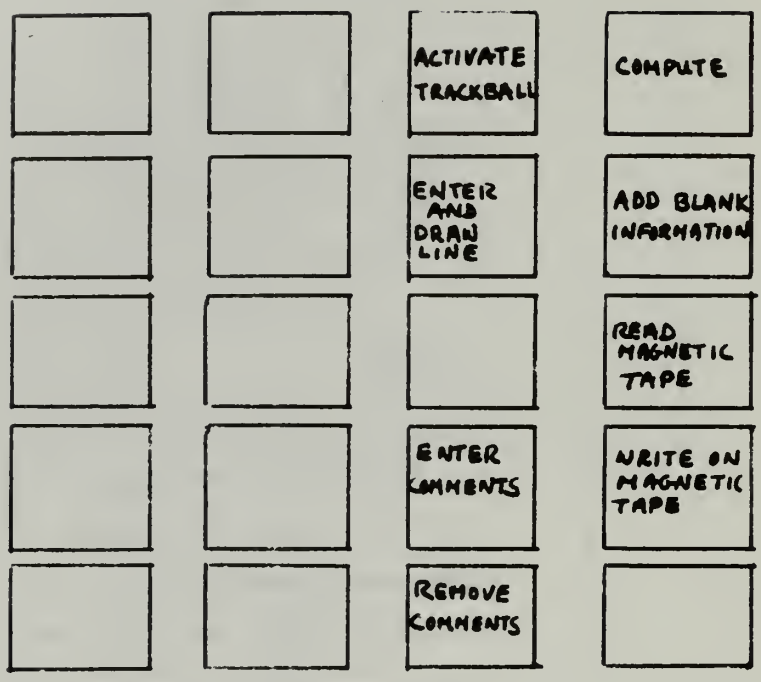

FIGURE 38

FUNCTIONAL DIAGRAM OF KEYBOARD II
ON THE DD 65 CONSOLE

BIBLIOGRAPHY

1. Granino Arthur Korn. Electronic Analog Computers (d. c. Analog Computers) (second edition; New York: McGraw Hill Company, 1956), pp. 290-299.

2. T. Miura and others. A New Diode Function Generator (Institute of Radio Engineers Transactions Electronic Computers, June, 1957) Vol. EC-6.

3. E. J. Galli. How Diodes Generate Functions (Control Engineering, March, 1959), pp. 109-113.

APPENDIX I

FORTRAN PROGRAM FOR THE DESIGN OF SYMBOLS

```
COOP,0595,TYNAN JM,S/1S/2S,15,10000.
FTN,L,E.
      PROGRAM DSINET
      DIMENSIONNXY(60),IRPOS(30),IRNEG(3 ),IVPOS(30),IVNEG(30),ITEMP(30)
      DO 1I=1,30
      NXY(2*I-1)=0
      NXY(2*I)=0
      ITEMP(I)=0
      IRPOS(I)=0
      IRNEG(I)=0
      IVPOS(I)=0
    1 IVNEG(I)=0
      IFLAG=-1
      DO 530 K=1,2
      NXY IS A TABLE OF X SPACE Y SPACE ETC. FORMAT
      READ 2,(NXY(I),I=1,28,2)
    2 FORMAT((7I10))
      NOW THE COMPUTATIONS FOR X OR Y WILL BE DONE
      IFLAG = IFLAG+1
      IT = 0
      ID = 0
      IV = 0
      ITBLPTR = 1
      KPNTR = 1
    3 JPNTR = 1
      DO 5 I=1,60,2
      IF(NXY(I)+NXY(I+2))5,6,5
    5 CONTINUE
    6 ITBLPTR = I-8
    8 ISENS = 25
      ISENS IS OUTPUT SENSITIVITY IN INPUT UNITS/MILLIAMETER
      IVMAX=20000
      IVMAX IS MAXIMUM INPUT VOLTAGE IN MILLIVOLTS
      RT = IDELTAD*VMAX*ISENS/(ID*CHANGE IN NXY)
   10 DO 11 I=1,I TBLPTR,4
      IDELX = NXY(I+4)-NXY(I)
```

```
      IDELY = NXY(I+6)-NXY(I+2)
      ISQUARE = IDELX**2+IDELY**2
      ITEMP(I+4)=ISQUARE/2
      DO 101 J=1,20
      INT1=ISQUARE-ITEMP(I+4)**2
      INT2=2*ITEMP(I+4)
      ICORRECT=INT1/INT2
  101 ITEMP(I+4)=ITEMP(I+4)+ICORRECT+2*(INT1-ICORRECT*INT2)/INT2
      IT = IT +ITEMP(I+4)
      ID = IT
      ITEMP(I+4) =IT
      IF (IFLAG)100,11,100
  100 ITEMP(I+6)=ITEMP(I+4)
   11 CONTINUE
      IVAR = 1
      IF (IFLAG)12,13,12
   12 ITBLPTR = ITBLPTR + 2
      IVAR = 3
C     NOW TO COMPUTE IR NEG AND IR POS
   13 DO 41 I=IVAR,ITBLPTR,4
      ITEMP1=NXY(I+4)-NXY(I)
      IF(ITEMP1)131,132,132
  131 SINFLAG = -1
      ITEMP1 =ITEMP1*SINFLAG
      GO TO 133
  132 SINFLAG =1
  133 IRT2 = (ITEMP(I+4)-ITEMP(I))*IVMAX*ISENS/(ID*999999)
      IF(ITEMP1-IRT2)32,32,14
   14 IRT2 = (ITEMP(I+4)-ITEMP(I))*IVMAX*ISENS/(ID*(NXY(I+4)-NXY(I)))
      GO TO 45
   32 IRT2=999999*SINFLAG
   45 IF(I-IVAR)33,44,33
   44 IRT = IRT2
      GO TO 37
C     JPNTR IS INDEX FOR POS QUANTITIES
C     KPPTR IS INDEX FOR NEG QUANTITIES
```

```
 33 IRD = IRT1-IRT2
    IF(IRD)34,331,331
 34 SINFLAG=-1
    IRD=IRD*SINFLAG
    GO TO 332
331 SINFLAG=1
332 IRT = IRT2*IRT1/999999
    IF(IRD-IRT)35,35,36
 35 IRT = 999999*SINFLAG
    GO TO 37
 36 IRT = IRT2*IRT1/(IRT1-IRT2)
    IRT = IRT - ((IRT*ITEMP(I))/ID)
 37 IF(IRT)38,39,39
C      MAKE TABLES OD POSITIVE AND NEGATIVE VOLTAGES
 38 IRNEG(JPNTR)=-IRT
    IVNEG(JPNTR) =IVMAX*ITEMP(I)/ID +IV
    JPNTR = JPNTR + 2
    GO TO 40
 39 IRPOS(KPNTR) = IRT
    IVPOS (KPNTR)=IVMAX*ITEMP(I)/ID+IV
    KPNTR = KPNTR + 2
 40 IRT1 = IRT2
 41 CONTINUE
    JPNTR = JPNTR - 2
    KPNTR = KPNTR - 2
    DO 50 J=1,KPNTR,2
    I1=IVMAX*IRPOS(J)/(IVMAX-IVPOS(J))
    I2=IVMAX*IRPOS(J)/IVPOS(J)
    IF(IVPOS(J))806,806,802
806 I2=999999
    GO TO 803
802 I2=I2-60
803 IRPOS(J)=I1
 50 IVPOS(J)=I2
    DO 51 J=1,JPNTR,2
    I1=IVMAX*IRNEG(J)/(IVMAX-IVNEG(J))
```

```
      I2=IVMAX*IRNEG(J)/(IVNEG(J))
      IF(IVNEG(J))606,606,102
  606 I2=999999
      GO TO 601
  102 I2 = I2-60
  601 IRNEG(J)=I1
   51 IVNEG(J)=I2
      IF (IFLAG)700,700,701
  700 PRINT 702
      GO TO 704
  701 PRINT 703
  702 FORMAT(1H1,12X,47HTHESE ARE THE X FUNCTION GENERATOR POT SETTINGS)
  703 FORMAT(///12X,47HTHESE ARE THE Y FUNCTION GENERATOR POT SETTINGS)
  704 PRINT 52,(IRNEG(I),IVNEG(I),I=1,JPNTR,2)
   52 FORMAT(//(22X,6HRNEG1=,I6,6X,6HRNEG2=,I6))
  530 PRINT 53,(IRPOS(I),IVPOS(I),I=1,KPNTR,2)
   53 FORMAT(//(22X,6HRPOS1=,I6,6X,6HRPOS2=,I6))
      END
      END
         FINIS
-EXECUTE.
         1          1          0        301        190        301        240
        25        240        190        190        140          0        140
```

THESE ARE THE X FUNCTION GENERATOR POT SETTINGS

RNEG1=	17[illegible]68	RNEG2=	999999
RNEG1=	1912	RNEG2=	147[illegible]
RNEG1=	695	RNEG2=	40[illegible]
RNEG1=	834	RNEG2=	278
RNEG1=	1912	RNEG2=	465
RPOS1=	565	RPOS2=	1[illegible]36

THESE ARE THE Y FUNCTION GENERATOR POT SETTINGS

RNEG1=	566	RNEG2=	1[illegible]39
RNEG1=	789	RNEG2=	571
RNEG1=	2[illegible]93	RNEG2=	1[illegible]86
RPOS1=	566	RPOS2=	999999
RPOS1=	1912	RPOS2=	743
RPOS1=	8[illegible]	RPOS2=	16[illegible]

APPENDIX II

ASSEMBLY LANGUAGE PROGRAM FOR THE DESIGN OF SYMBOLS, INCLUDING GRAPHICAL DISPLAY

```
      0100  consol  org   100          console control rtne
0100  7500  wait    exf
0101  7040                7040         select status dd65
0102  7600          ina                A
0103  0201          lpn   1            was keybboard 2 hit
0104  6404          zjb   wait         if not go wait
0105  7500          exf
0106  7120                7120         sel input from keyboard 2
0107  7600          ina                B
0110  0277          lpn   77
0111  4205          stf   code
0112  2200          ldf
0113  7000                7000         C
0114  1602          scf   code
0115  4201          stf   1
0116  0000  code            0          D jump to appropriate driver
      0000          end
```

```
      0100            org   100       trackball driver routine
0100  0101  tb        pta             |A
0101  7041            jpi   trkbal    |
0102  7500            exf
0103  7040            ina   7040      | select dd65 status
0104  7600                            |B
0105  0201            lpn   1         | was keyboard 2 hit
0106  6002            zjf   delay     | if not go delay
0107  7044            jpi   consol
0110  2600  delay     lcf             |
0111  1750                  1750      |
0112  4077            std   cntr      |C
0113  5477            aod   cntr      |
0114  6501            nzb   1         |
0115  6415            zjb   tb        |
      0044  consol    equ   44
      0041  trkbal    equ   41
      0077  cntr      equ   77
      0000            end
```

```
      0100  trkbal  org   100
0100  0602         adn   2
0101  4233         stf   link     | A
0102  7500         exf            |
0103  7102                 7102   |
0104  7600         ina            | B
0105  0702         sbn   2        |   read trackball coordinates and
0106  4010         std   x        |   correct for offset
0107  7500         exf            |
0110  7104                 7104   |
0111  7600         ina            | C
0112  0704         sbn   4        |
0113  4011         std   y        |
0114  0400         ldn   0        |
0115  4012         std   tubvec   |   set up arguments for print
0116  2200         ldf            |
0117  3000                 3000   |
0120  4013         std   size     |
0121  2200         ldf            |
0122  1776                 1776   | D
0123  4014         std   m        |
0124  0460         ldn   60       |
0125  4015         std   loc      |
0126  2200         ldf            |
0127  4677                 4677   |   code for letter o
0130  4060         std   60       |
0131  0101         pta            |
0132  7043         jpi   print    |
0133  7101         jfi   1        | E
0134  0000  link                  |
      0010  x      equ   10
      0011  y      equ   11
      0012  tubvec equ   12
      0013  size   equ   13
      0014  m      equ   14
      0015  loc    equ   15
      0043  print  equ   43
      0000         end
```

```
      0173  drw     org   173
0173  0620          adn   20
0174  4262          stf   arg        store link address
0175  2035          ldd   tblpt   |  A
0176  6117          nzf   oldhat  |
0177  2243          ldf   setfil
0200  4036          std   eofpt   |
0201  2263          ldf   neg     |  B
0202  4035          std   tblpt   |
0203  2200          ldf
0204  7400                7400    |
0205  4075          std   temp    |
0206  2200  blank   ldf             |
0207  2020                2020    |  C
0210  4175          sti   temp    |
0211  5475          aod   temp    |
0212  1600          scf           |
0213  7477                7477    |
0214  6506          nzb   blank   |
0215  0101  oldhat  pta             |
0216  7041          jpi   trkbal  |  D
0217  2010          ldd   x       |
0220  4135          sti   tblpt   |
0221  0110          ls3           |
0222  6204          pjf   4       |
0223  2241          ldf   neg     |
0224  1410          scd   x       |
0225  4135          sti   tblpt   |
0226  5435          aod   tblpt   |
0227  5435          aod   tblpt   |  E  move table pointer to y
0230  2011          ldd   y       |
0231  4135          sti   tblpt   |
0232  0110          ls3           |
0233  6204          pjf   4       |
0234  2230          ldf   neg     |
0235  1411          scd   y       |
0236  4135          sti   tblpt   |     x and y are entered in
0237  5435          aod   tblpt   |
0240  5435          aod   tblpt   |     pointer set for end of table
0241  2600          lcf           |
0242  4000  setfil        4000    |
0243  4076          std   cntr    |
0244  7500          exf           |  F
0245  7020                7020    |
0246  7400          otn           |
0247  5476          aod   cntr    |
0250  6502          nzb   2       |     dd65 is erased
0251  2200          ldf           |
0252  7004                7004    |
0253  1435          scd   tblpt   |
```

```
0254  6007          zjf   exit   |G exit if first table entry
0255  0101          pta          |
0256  0000  arg                   |H
0257  2035          lda   tblpt  |
0260  6003          zjf   exit   |J
0261  0101          pta          |
0262  7046          jpi   dis    |K
0263  7021  exit    jpi   tb
0264  7000  neg           7000
      0010  x       equ   10
      0011  y       equ   11
      0012  tubvec  equ   12
      0021  tb      equ   21
      0046  dis     equ   46
      0035  tblpt   equ   35
      0036  eofpt   equ   36
      0041  trkbal  equ   41
      0076  cntr    equ   76
      0075  temp    equ   75
      0015  loc     equ   15
      0000          end
```

```
      0257  linel   org   257
0257  0602          adn   2          | A
0260  4062          std   link       |
0261  5436          aod   eofpt      |
0262  0510          lcn   10         |
0263  3035          add   tblpt      |
0264  4035          std   tblpt      |
0265  2135          ldi   tblpt      |
0266  4010          std   xl         |
0267  1200          lpf              |
0270  0777                777        |
0271  4136          sti   eofpt      |
0272  5436          aod   eofpt      |
0273  5435          aod   tblpt      |
0274  5435          aod   tblpt      |
0275  2135          ldi   tblpt      |
0276  4011          std   yl         |
0277  1200          lpf              |
0300  0777                777        |
0301  4136          sti   eofpt      |
0302  0502          lcn   2          |
0303  5036          rad   eofpt      | B
0304  5435          aod   tblpt      |
0305  5435          aod   tblpt      |
0306  2135          ldi   tblpt      |
0307  4070          std   x2         |
0310  5435          aod   tblpt      |
0311  5435          aod   tblpt      |
0312  2135          ldi   tblpt      |
0313  4071          std   y2         |
0314  5435          aod   tblpt      |
0315  5435          aod   tblpt      |
0316  2200          ldf              |
0317  2000                2000       |   tubvec set for left tube
0320  4136          sti   eofpt      |
0321  0403          ldn   3          |
0322  5036          rad   eofpt      |
0323  2410          lcd   xl         |
0324  3070          add   x2         |
0325  4066          std   a          |
0326  6206          pjf   6          |
0327  2466          lcd   a          |
0330  4066          std   a          | C
0331  0404          ldn   4          |
0332  4077          std   icode      |
0333  6103          nzf   3          |
0334  0400          ldn   0          |
0335  4077          std   icode      |
```

```
0336  2411          lcd   yl
0337  3071          add   y2
0340  4067          std   b
0341  6206          pjf   6
0342  2467          lcd   b          D
0343  4067          std   b
0344  0406          ldn   6
0345  4076          std   jcode
0346  6103          nzf   3
0347  0402          ldn   2
0350  4076          std   jcode
0351  3077          add   icode
0352  0111          ls6
0353  0110          ls3
0354  0103          ls2              circulate 11
0355  4075          std   iboth
0356  0303          scn   3          E
0357  6105          nzf   5
0360  2077          ldd   icode
0361  6103          nzf   3
0362  0404          ldn   4
0363  5075          rad   iboth
0364  2067          ldd   b
0365  1200          lpf
0366  0774                774        chop off bottom two bits
0367  0103          ls2
0370  0103          ls2
0371  0111          ls6              circulate 10
0372  4067          std   b          b now no. of y vectors
0373  2066          ldd   a
0374  1200          lpf
0375  0774                774        F  chop off bottom two bits
0376  0103          ls2
0377  0103          ls2
0400  0111          ls6              circulate 10
0401  4066          std   a          a now no. of x vectors
0402  3067          add   b
0403  6103          nzf   3
0404  4035          std   tblpt
0405  6034          zjf   exitl      G  exit if input zero
0406  2066          ldd   a
0407  3467          sbd   b
0410  6206          pjf   6
0411  2066          ldd   a
0412  4074          std   nrshrt     H  x has least vectors
0413  2067          ldd   b
0414  4073          std   nrlong        y has most vectors
```

```
0415  6107          nzf   7          |    note mostly equals jcode
0416  2077          ldd   icode      |    line inclined less than 45 degrees
0417  4076          std   mostly     |    to x axis
0420  2067          ldd   b          |
0421  4074          std   nrshrt     | J  y has least vectors
0422  2066          ldd   a          |
0423  4073          std   nrlong     |    x has most vectors
0424  2473          lcd   nrlong     |
0425  4077          std   limit      |    limit equals -no. of most vectors
0426  2074          ldd   nrshrt     |
0427  4072          std   switch     |
0430  0440          ldn   40         |
0431  4071          std   intbit     |
0432  2071  return    ldd   intbit   |
0433  6107          nzf   clear    | K
0434  2200          ldf              |
0435  7700          7700           |    store 7700 at end of file
0436  4136          sti   cofpt    | L
0437  5436          aod   eofpt    |
0440  6147          nzf   exit       |
0441  6046  exitl   zjf   exit       |
0442  0400  clear   ldn   0          |
0443  4066          std   lr         |    set lr for right
0444  2073  gnx     ldd   nrlong     |
0445  3472          sbd   switch   | M
0446  3472          sbd   switch     |
0447  6210          pjf   10         |    test to determine code for
0450  2075          ldd   iboth    |    next vector
0451  4065          std   move     | N  45 degree vector chosen
0452  2074          ldd   nrshrt     |
0453  3473          sbd   nrlong   | P
0454  5072          rad   switch     |
0455  6006          zjf   6          |
0456  6105          nzf   5          |
0457  2076          ldd   mostly     |
0460  4065          std   move     | Q  a vector from mostly chosen
0461  2074          ldd   nrshrt     |
0462  5072          rad   switch     |
0463  5477          aod   limit      |    check off one vector from limit
0464  6102          nzf   2          |
0465  4071          std   intbit   | R  limit reached
0466  2065          ldd   move       |
0467  1471          scd   intbit     |    form vector code
0470  4065          std   move       |
0471  2466  testlr    lcd   lr       |
0472  4066          std   lr       | S  |
0473  6107          nzf   left       |
0474  2065          ldd   move       |
0475  1463          scd   buf      | T  |
0476  4136          sti   eofpt      |    assembleone word
```

```
0477  5436          aod   eofpt
0500  6546          nzb   return
0501  6447          zjb   return
0502  2065  left    ldd   move
0503  0111          ls6
0504  4063          std   buf
0505  6441          zjb   gnx
0506  6542          nzb   gnx
0507  7062  exit    jpi   link
      0077  icode   equ   77
      0076  jcode   equ   76
      0075  iboth   equ   75
      0074  nrshrt  equ   74
      0073  nrlong  equ   73
      0076  mostly  equ   76
      0077  limit   equ   77
      0072  switch  equ   72
      0071  intbit  equ   71
      0066  a       equ   66
      0067  b       equ   67
      0066  lr      equ   66
      0065  move    equ   65
      0063  buf     equ   63
      0062  link    equ   62
      0010  xl      equ   10
      0070  x2      equ   70
      0011  yl      equ   11
      0071  y2      equ   71
      0036  eofpt   equ   36
      0035  tblpt   equ   35
      0015  loc     equ   15
      0012  tubvec  equ   12
      0000          end
```

U temporarily store left half word

```
      0516          org   516
0516  0602  print   adn   2       | A  clear 160 upper lower half ind
0517  4007          std   7          |  set linkage
0520  0400          ldn             |  clear 160 upper lower half ind
0521  4072          std   ul        | B  end message flag
0522  4070          std   flag      |    and
0523  4063          std   nflag     |    clear multiple vector flag
0524  2012          ldd   tubvec       load tube vector argument
0525  3243          adf   lthous    |
0526  4244          stf   desig     |     generate designator bits
0527  2010          ldd   x         |     b coordinate
0530  1241          lpf   mask      |     mask right 9 bits
0531  1413          scd   size      |     merge size argument bits
0532  4067          std   buf1      | C   hold for output
0533  2014          ldd   m         |     dd65 memory address
0534  4065          std   buf3      |     hold for output
0535  2011          ldd   y         |     y coordinate
0536  1233          lpf   mask      |
0537  1633          scf   desig     |     merge desig bits
0540  4064          std   buf4      |
0541  6106          nzf   short        go add 2 characters
0542  0503  string  lcn   3       |
0543  4074          std   wrdctr    | D  set to count 12 bit word
0544  0507          lcn   7         |
0545  4075          std   bytctr    |    set to count 6 bit byts
0546  6105          nzf   clear        unconditional jump
0547  0502  short   lcn   2       |
0550  4074          std   wrdctr    |
0551  0502          lcn   2         |
0552  4075          std   bytctr    | E  short two char seq. for desig w
0553  0400  clear   ldn             |
0554  4071          std   lr           clear dd65 left right indicator
0555  2070  rnx     ldd   flag      |  read next character routine
0556  6003          zjf   3         | F
0557  0400          ldn             |
0560  6067          zjf   term      |  fill blank if flag set
0561  2063          ldd   nflag     |
0562  6030          zjf   normal    | G  check if mult vector flag set
0563  0376          scn   76        |    chek if flag being set
0564  6007          zjf   go        | H
0565  5463          aod   nflag     |    flag is set
0566  6124          nzf   normal    | J
0567  6023          zjf   normal       unconditional jump
```

```
0570  1000  lthous        1000
0571  0777  mask          777
0572  0000  desig         0
0573  2072  go      ldd   ul        K L  set flag routine
0574  6310          njf   upr       |
0575  2115          ldi   loc       |
0576  0111          ls6             |
0577  0277          lpn   77        |
0600  4063          std   nflag     N
0601  0563          lcn   nflag     |
0602  4063          std   nflag     |
0603  6107          nzf   normal    |
0604  2115  upr     ldi   loc       |
0605  0277          lpn   77        |
0606  4063          std   nflag     |
0607  2463          lcd   nflag     M
0610  4063          std   nflag     |
0611  5415          aod   loc       |
0612  2115  normal  ldi   loc       |   pick up pair of characters
0613  4077          std   temp      |
0614  2063          ldd   nflag     |
0615  6103          nzf   3         |
0616  2472          lcd   ul            change upper lower indicator
0617  4072          std   ul
0620  2072          ldd   ul
0621  6010          zjf   lower
0622  2077          ldd   temp      |
0623  0111          ls6             |   circulate 6 places
0624  4077          std   temp      M
0625  6107          nzf   maskof    |
0626  6006          zjf   maskof    |   uncond. jump to maskof right 6
0627  6552  rnxl    nzb   rnx
0630  6466  jump    zjb   string
0631  2063  lower   ldd   nflag     |
0632  6102          nzf   maskof    N  loc unchanged for multi vector
0633  5415          aod   loc       |  increment 160 memory loc.
0634  2077  maskof  ldd   temp      |
0635  0277          lpn   77        |
0636  4077          std   temp      O
0637  0376          scn   76        |
0640  6104          nzf   tstend    |   test for multi vector
0641  2077          ldd   temp      |
0642  4063          std   nflag     P
0643  6566          nzb   rnx       |
0644  2077  tstend  ldd   temp      |
0645  0377          scn   77        |
```

```
0646  6104          nzf   contin     sense 77 code
0647  4077  term    std   temp
0650  0500          lcn               Q
0651  4070          std   flag
0652  2471  contin  lcd   lr
0653  4071          std   lr         R change lr indicator
0654  6116          nzf   left
0655  2474          lcd   wrdctr
0656  3232          adf   outbuf     count outbuf backwards
0657  4073          std   index      form indexer
0660  2077          ldd   temp       S
0661  1476          scd   buf
0662  4173          sti   index      pack output word
0663  5474          aod   wrdctr
0664  6111          nzf   cycle      loop till seventh
0665  2012          ldd   tubvec
0666  0111          ls6               T
0667  4077          std   temp
0670  6616          pjb   contin     merge tubvec bits
0671  6717          njb   contin
0672  2077  left    ldd   temp
0673  0111          ls6               U circulate 6
0674  4076          std   buf
0675  5475  cycle   aod   bytctr
0676  6547          nzb   rnxl       V
0677  0402          ldn   2
0700  5014          rad   m
0701  7506          exf   update     W
0702  7306          out   outbuf
0703  0070                70
0704  2070          ldd   flag
0705  6455          zjb   jump       X
0706  7007          jpi   7
0707  7020  update        7020
0710  0064  outbuf      64
      0077  temp    equ   77
      0076  buf     equ   76
      0075  bytctr  equ   75
      0074  wrdctr  equ   74
      0073  index   equ   73
      0072  ul      equ   72
      0071  lr      equ   71
      0070  flag    equ   70
```

```
0067  buf1     equ   67
0066  buf2     equ   66
0065  buf3     equ   65
0064  buf4     equ   64
0063  nflag    equ   63
0010  x        equ   10
0011  y        equ   11
0012  tubvec   equ   12
0013  size     equ   13
0014  m        equ   14
0015  loc      equ   15
0000           end
```

```
      0750              org   750          displays the file
0750  0602  dis         adn   2       |A
0751  4264              stf   link      |
0752  2611  erase       lcf   setfil    |
0753  4076              std   temp      |
0754  7500              exf             |
0755  7020                    7020      |B
0756  7400              otn             |
0757  5476              aod   temp      |
0760  6502              nzb   2         |
0761  4014              std   m         |
0762  2200              ldf             |
0763  4000  setfil      4000            |
0764  4015              std   loc       |
0765  1436              scd   eofpt     |C
0766  6046              zjf   exit      |
0767  2200              ldf             |
0770  3600                    3600      |
0771  4037              std   psnpt     |
0772  4076              std   temp      |
0773  0400  clear       ldn             |D
0774  4176              sti   temp      |
0775  5476              aod   temp      |
0776  1415              scd   loc       |
0777  6504              nzb   clear     |  clear the position table
1000  2115  loop        ldi   loc       |
1001  4012              std   tubvec    |
1002  5415              aod   loc       |E
1003  2115              ldi   loc       |
1004  4010              std   x         |
1005  0110              ls3             |
1006  1200              lpf             |
1007  7700                    7700      |
1010  4076              std   temp      |
1011  5415              aod   loc       |
1012  2115              ldi   loc       |
1013  4011              std   y         |F  x,tubvec and y args now in print
1014  0110              ls3             |
1015  0111              ls6             |
1016  0277              lpn   77        |
1017  1476              scd   temp      |
```

```
1020  4137          sti   psnpt |     xy now in position table
1021  5437          aod   psnpt |
1022  2015          ldd   loc   | F   record starts two before y
1023  0702          sbn   2     |
1024  4137          sti   psnpt |     position table contains address
1025  5437          aod   psnpt |     of start record
1026  5415          aod   loc         loc set for beginning of string
1027  0101          pta               |
1030  7043          jpi   print | G   |
1031  5415          aod   loc         |
1032  1436          lsd   eofpt | H   |
1033  6533          nzb   loop  |     |
1034  7101  exit    jfi   link        |
1035  0000  link                      |
      0037  psnpt   equ   37          |
      0076  temp    equ   76          |
      0015  loc     equ   15          |
      0014  m       equ   14          |
      0012  tubvec  equ   12          |
      0010  x       equ   10          |
      0011  y       equ   11          |
      0043  print   equ   43          |
      0036  eofpt   equ   36          |
      0000          end               |
```

```
      1050          org   1050          routine to enter characters from
1050  0101  enter    pta                 dd65 keyboard 1 into a file
1051  7041          jpi   trkbal   A     which starts at cell 4000 and
1052  0400          ldn   0             ends at eofpt stored in cell 36
1053  4060          std   ul
1054  4136          sti   eofpt         enter tubvec code in file
1055  5436          aod   eofpt
1056  2010          ldd   x
1057  4136          sti   eofpt    B    enter x into file
1060  5436          aod   eofpt
1061  2011          ldd   y
1062  4136          sti   eofpt         enter y into file
1063  7500  wait     exf
1064  7040                        7040  select dd65 status
1065  7600          ina
1066  0206          lpn   6        C
1067  6404          zjb   wait          wait until keyboard 1 or cr hit
1070  4061          std   code
1071  0204          lpn   4        D
1072  6146          nzf   exit          exit if cr hit
1073  2061          ldd   code
1074  0202          lpn   2
1075  6412          zjb   wait          was keyboard 1 hit
1076  7500          exf                 keyboard 1 was hit
1077  7140                        7140  select input from keyboard 1
1100  7600          ina
1101  4062          std   char
1102  2460  store    ldd   ul       E
1103  4060          std   ul            is this upper or lower
1104  6214          pjf   lower
1105  5436          aod   eofpt
1106  2062          ldd   char          is upper
1107  0111          lsb
1110  0320          scn   20
1111  4136          sti   eofpt    F    put char with eof mark in file
1112  5436          aod   eofpt         set end of file
1113  2216          ldf   neg
1114  4136          sti   eofpt
1115  5436          aod   eofpt
1116  0400          ldn   0
1117  6011          zjf   des           go to clear select and display
1120  2136  lower    ldi   eofpt        is a lower character
1121  0320          scn   20           clear eof mark from file word
1122  1462          scd   char         add new char to file word
1123  4136          sti   eofpt        put word back into file
1124  5436          aod   eofpt    G
1125  2204          ldf   neg
1126  4136          sti   eofpt         put eof mark into file
1127  5436          aod   eofpt
1130  7500  des      exf
                                  H
```

```
1151  7700  neg          7700  | H  clear dd65 select
1152  0101        pta
1153  7046        jpi    dis   | J  display file showing new character
1154  0502        lcn    2
1155  5036        rad    eofpt      reset file marker for new character
1156  0400        ldn    0
1157  6454        zjb    wait  |    wait for new character
1140  7500  exit  exf           |
1141  7140                7140  | K
1142  7600        ina           |
1143  0402        ldn    2          exit after carriage return
1144  5036        rad    eofpt
1145  7021        jpi    tb
      0060  ul    equ    60
      0061  code  equ    61
      0062  char  equ    62
      0046  dis   equ    46
      0044  consol equ   44
      0041  trkbal equ   41
      0036  eofpt equ    36
      0021  tb    equ    21
      0010  x     equ    10
      0011  y     equ    11
      0000        end
```

```
      1150          org   1150        removes a string of
1150  0101  remove  pta                vectors or characters from
1151  7041          jpi   trkbal  | A  the computer file
1152  2010          ldd   x
1153  0110          ls3               shift x
1154  1200          lpf
1155  7700                7700        mask x
1156  4076          std   xy          store x
1157  2011          ldd   y
1160  0110          ls3               shift y
1161  0111          ls6
1162  0277          lpn   77          mask y
1163  1476          scd   xy          form xy coordinates
1164  4076          std   xy
1165  2200          ldf
1166  3600                3600
1167  4077          std   posit       initialize position table address
1170  2177  search  ldi   posit       search for a string with same
1171  1476          scd   xy    C     xy coords as trkbal
1172  6006          zjf   found D
1173  0402          ldn   2
1174  5077          rad   posit
1175  1437          scd   psnpt
1176  6506          nzb   search| H
1177  7021  exit    jpi   tb          exit after removing string or if
1200  5477  found   aod   posit       string not found
1201  2177          ldi   posit
1202  4014          sta   loc         put address of file to be removed in loc
1203  5477          aod   posit
1204  1437          scd   psnpt E     check if this is the last string
1205  6012          zjf   last
1206  5477          aod   posit
1207  2177          ldi   posit
1210  4075          std   next        put start of next file in next
1211  2175  moveup  ldi   next
1212  4114          sti   loc
1213  5414          aod   loc
1214  5475          aod   next  F
1215  1436          scd   eofpt
1216  6505          nzb   moveup
1217  2014  last    ldd   loc
1220  4036          std   eofpt       end of file pt is at loc
1221  0101          pta
1222  7046          jpi   dis   | G
1223  0400          ldn   0
1224  6425          zjb   exit
      0077  posit   equ   77
      0076  xy      equ   76
      0075  next    equ   75
```

```
0074  cntr     equ   74
0046  dis      equ   46
0044  consol   equ   44
0041  trkbal   equ   41
0037  psnpt    equ   37
0036  eofpt    equ   36
0021  tb       equ   21
0010  x        equ   10
0011  y        equ   11
0014  loc      equ   14
0000           end
```

```
     1250  comput  org   1250
1250 0504          lcn   4          computes resistance,bias voltage
1251 5035          rad   tblpt      values
1252 0400          ldn   0
1253 4012          std   dist
1254 4013          std   updist
1255 2200  table   ldf
1256 7000  addr          7000
1257 4050          std   index      A  index is table index
1260 0602          adn   2
1261 4052          std   index2
1262 0602          adn   2
1263 4054          std   index4
1264 0602          adn   2
1265 4056          std   index6
1266 2200          ldf
1267 7300                7300
1270 4015          std   tpt        all indices are now set
1271 0400          ldn   0
1272 4115          sti   tpt
1273 5415          aod   tpt
1274 0400          ldn   0
1275 4115          sti   tpt
1276 0403          ldn   3
1277 5015          rad   tpt
1300 2200          ldf
1301 0400                400      B
1302 4002          std   vmax
1303 0400  loop    ldn
1304 4061          std   buf1
1305 4063          std   buf3       C
1306 4011          std   umin
1307 2550          lci   index
1310 3154          adi   index4
1311 4060          std   delx       form delta x
1312 6203          pjf   3
1313 2461          lcd   buf1       convert to 22 bit arith format
1314 4061          std   buf1
1315 2552          lci   index2
1316 3156          adi   index6
1317 4062          std   dely       D  form delta y
1320 6203          pjf   3
1321 2463          lcd   buf3       convertto 22 bit arith format
1322 4063          std   buf3
1323 0101          pta
1324 7040          jpi   arith
1325 0031                31
1326 0060                delx
1327 0060                delx
1330 0060                delx       delta x is now squared
```

```
1331  0031                    31
1332  0062                    dely
1333  0062                    dely
1334  0062                    dely      D   delta y is now squared
1335  0001                    1
1336  0060                    delx
1337  0062                    dely
1340  0060                    square        form sum of squares
1341  0400          ldn       0
1342  6003          zjf       root
1343  0000  xyflag
1344  6541  loopl   nzb       loop
1345  0101  root    pta
1346  7047          jpi       sroot     E   root goes to tpt location
1347  0101          pta
1350  7040          jpi       arith
1351  0001                    1
1352  0015                    tpt
1353  0012                    dist
1354  0012                    dist      F   distance is old distance plus
1355  2012          ldd       dist          square root sqare
1356  4115          sti       tpt           tpt becomes the present distance
1357  5415          aod       tpt
1360  2013          ldd       updist
1361  4115          sti       tpt           transferred top half word
1362  0404  ini     ldn       4             now to increase indices
1363  5050  instl   rad       index
1364  5701          aob       instl
1365  0207          lpn       7
1366  6504          nzb       ini       G   check if all indices are increased
1367  0403          ldn       3
1370  5015          rad       tpt           tpt now increased by four
1371  0510          lcn       10
1372  5307          rab       instl         instl now reset
1373  2050          ldd       index
1374  1435          scd       tblpt     H   check if index at end of table
1375  6531          nzb       loopl
1376  2200          ldf
1377  7000  tad               7000          finished dist time computations
1400  4051          std       indexl        set for first x
1401  2736          lcb       xyflag
1402  4337          stb       xyflag
1403  6105          nzf       exx       J
1404  0402          ldn       2         K   is a y calculation
1405  5051          rad       indexl    L   set for first y
1406  0402          ldn       2
1407  5035          rad       tblpt         set for last y
1410  2311  exx     ldb       tad           tablpt set for last x
1411  3230          adf       hund          if an x calculation
```

```
1412  4057          std   index7 |   rpos table index
1413  3226          adf   hund   |
1414  4053          std   index3 |   rneg table index
1415  3224          adf   hund   |
1416  4055          std   index5 |   t table index
1417  0404          ldn   4      |
1420  3051          add   indexl |
1421  4050          std   index      index set for second x or y
1422  6102          nzf   2
1423  0777  arg2          777    | M
1424  2301  begin   ldb   arg2
1425  4000          std   sens
1426  0400          ldn   0      |
1427  4001          std   1      |
1430  4005          std   udiff  | N
1431  4067          std   buf7   |
1432  4011          std   11     |   clear upper part of double precision
1433  2551          lci   indexl     cells
1434  3150          adi   index  |
1435  4004          std   diff   |   forms delta x or delta y
1436  6304          njf   comp   |
1437  0401          ldn   1      |
1440  6207          pjf   load   |   unconditional jump
1441  0100  hund          100    |
1442  2404  comp    lcd   diff   |   compliment diff to make positive
1443  4004          std   diff   |
1444  0500          lcn          | O
1445  4067          std   buf7   |
1446  0501          lcn   1      |
1447  4066  load    std   sinflg |   sinflag contains the sign of
1450  2307          ldb   hund   |   delta x or delta y
1451  4010          std   min    |   this is the minimum r allowed
1452  2200          ldf          |
1453  3777                3777   |
1454  4016          std   max    |   this is the maximum
1455  0403          ldn   3      |
1456  4017          std   17     |   r allowed
1457  2015          ldd   tpt
1460  0704          sbn   4
1461  4027          std   dtl
1462  2004          ldd   diff
1463  6021          zjf   big
1464  0101          pta          |
1465  7020          jpi   fixdpt |   ans goes to rt2
1466  0101          pta          |
1467  7040          jpi   arith  |
1470  0002                2      |
1471  0016                max    | P
1472  0064                rt2    |
1473  0004                testl  |   testl is difference between max r
1474  0002                2      |   and actual r
1475  0064                rt2    |
1476  0010                min    |
1477  0000                test2  |   test2 similar but using min
```

```
1500  2005         ldd   testlu |  Q   check sign of testl
1501  6213         pjf   test   |
1502  6302         njf   big    |
1503  6557  int    nzb   begin
1504  0101  big    pta                r is too big
1505  7040         jpi   arith  |
1506  0031               31     |
1507  0016               max    |  R
1510  0066               sinflg |
1511  0064               rt2    |
1512  0401         ldn   1
1513  6221         pjf   okay         unconditional jump
1514  2001  test   ldd   test2u |  S   test sign of test2
1515  6211         pjf   sign   |
1516  0101         pta
1517  7040         jpi   arith  |     r is too small
1520  0031               31     |
1521  0010               min    |  T
1522  0066               sinflg |
1523  0064               rt2    |
1524  0401         ldn   1
1525  6207         pjf   okay         unconditional jump
1526  0101  sign   pta              |
1527  7040         jpi   arith      |
1530  0031               31         |
1531  0064               rt2        |  U
1532  0066               sinflg     |
1533  0064               rt2        |     r gets proper sign
1534  2051  okay   ldd   indexl       test if this is the first
1535  0277         lpn   77           computation
1536  0703         sbn   3      |  Y
1537  6210         pjf   rcalc  |
1540  2004         ldd   rt2          is first calculation
1541  4066         std   r      |
1542  2065         ldd   buf5   |  W
1543  4061         std   buf1   |
1544  0001         ldn   1
1545  6265         pjf   tblstr       r is equal to rt2
1546  6543  over   nzb   int
1547  0101  rcalc  pta
1550  7040         jpi   arith  |
1551  0031               31     |
1552  0062               rtl    |  X
1553  0064               rt2    |
1554  0060               buf    |
```

```
1555  0002                 2
1556  0062                 rtl
1557  0064                 rt2
1560  0004                 diff
1561  2004          ldd    diff
1562  6034          zjf    tops      X
1563  0101          pta
1564  7040          jpi    arith
1565  0041                 41
1566  0060                 buf
1567  0004                 diff
1570  0060                 r              r eqals rtl rt2 over rtl minus rt2
1571  2461          lcd    bufl           check sign of r
1572  6310          njf    alrite
1573  4061          std    bufl
1574  0501          lcn    1
1575  4066          std    sinflg    Y    r is negative
1576  0500          lcn
1577  4067          std    buf7
1600  2460          lcd    r
1601  4060          std    r              r is complemented
1602  0101  alrite  pta
1603  7040          jpi    arith
1604  0002                 2
1605  0061                 rtl
1606  0062                 r
1607  0001                 testl     Z
1610  0002                 2
1611  0060                 r
1612  0010                 min
1613  0000                 test2          have formed limit value test
1614  2005          ldd    testlu    AA   numbers
1615  6207          pjf    try            test if r too big
1616  2016  tops    ldd    max            r is too big
1617  4060          std    r
1620  2017          ldd    umax      BB
1621  4061          std    bufl
1622  0401          lcn    1
1623  6207          pjf    tblstr         unconditional jump
1624  2001  try     ldd    test2u         see if r is less than min
1625  6215          pjf    tblstr    CC
1626  2017          ldd    min
1627  4060          std    r
1630  2011          ldd    umin      DD
1631  4061          std    bufl
1632  2052  tblstr  ldd    index3
1633  4015          std    store
```

```
1634  2000          ldd   sinflg |
1635  6014          njf   change |EE negative flag indicates that r
1636  6202          pjf   direct |
1637  6571  loop3    nzb   over   |
1640  2057  direct   ldd   index7|   is to be stored in rneg table
1641  4053          std   index3 |
1642  0504          lcn   4      |EE
1643  5010          rad   store  |
1644  6307          njf   trnsfr|   transfer after inhibit index 3
1645  0504  change   lcn   4
1646  5057          rad   index7|
1647  2466          lcd   sinflg|
1650  4066          std   sinflg|FF
1651  2467          lcd   buf7  |
1652  4067          std   buf7  |
1653  0101  trnsfr   pta
1654  7040          jpi   arith
1655  0031                3l   |
1656  0060                r    |GG
1657  0066                sinflg|
1660  0053                index3|
1661  0402          ldn   2
1662  5053          rad   index3
1663  0101          pta
1664  7040          jpi   arith |
1665  0031                3l    |
1666  0002                vmax  |
1667  0055                index5|
1670  0062                buf2  |HH
1671  0041                41    |
1672  0062                buf2  |
1673  0012                dist  |
1674  0053                index3|
1675  2050          ldd   index
1676  0277          lpn   77
1677  1640          scf   blank
1700  4070          std   temp
1701  2053          ldd   index3
1702  3236          adf   quart
1703  4071          std   templ
1704  2170          ldi   temp
1705  4171          sti   templ
1706  2010          ldd   store
1707  4053          std   index3
1710  2064  fini    ldd   rt2      transfer rt2
1711  4062          std   rtl
```

```
1712  2065          ldd   buf5
1713  4063          std   buf3
1714  0404  redo    ldn   4
1715  5050  in      rad   index
1716  5701          aob   in
1717  0207          lpn   7
1720  6504          nzb   redo
1721  0510          lcn   10
1722  5305          rab   in
1723  2051          ldd   index1
1724  1435          scd   tblpt
1725  6566          nzb   loop3
1726  2100          ldi
1727  1344                1344
1730  6103          nzf   exit
1731  0502          lcn   2
1732  5035          rad   tblpt
1733  0404  exit    ldn   4
1734  5035          rad   tblpt
1735  7101          jfi   display
1736  2000  display       2000
1737  7400  blank       7400
1740  0400  quart       400
      0000  sens    equ   0
      0000  test2   equ   0
      0001  test2u  equ   1
      0002  vmax    equ   2
      0004  testl   equ   4
      0004  diff    equ   4
      0005  udiff   equ   5
      0005  testlu  equ   5
      0010  min     equ   10
      0010  store   equ   10
      0011  umin    equ   11
      0012  dist    equ   12
      0013  updist  equ   13
      0015  tpt     equ   15
      0016  max     equ   16
      0017  umax    equ   17
      0020  fixdpt  equ   20
      0026  dt      equ   26
      0027  dtl     equ   27
      0035  tblpt   equ   35
```

JJ

KK

LL check if x or y calculation

MMit was a y calc

NN

```
0047  sroot   equ   47
0050  index   equ   50
0051  index1  equ   51
0052  index2  equ   52
0053  index3  equ   53
0054  index4  equ   54
0055  index5  equ   55
0056  index6  equ   56
0057  index7  equ   57
0060  buf     equ   60
0060  delx    equ   60
0060  square  equ   60
0060  r       equ   60
0061  buf1    equ   61
0062  buf2    equ   62
0062  dely    equ   62
0062  rt1     equ   62
0063  buf3    equ   63
0064  buf4    equ   64
0064  rt2     equ   64
0065  buf5    equ   65
0066  buf6    equ   66
0066  sinflg  equ   66
0067  buf7    equ   67
0070  temp    equ   70
0071  temp1   equ   71
0040  arith   equ   40
0000          end
```

```
      2450            org   2450          read/write driver
2450  0500  rd        lcn                 read entry point
2451  4225            stf   type    | A    set up read code
2452  6111            nzf   wait
2453  0400  wr        ldn                 write entry point
2454  4222            stf   type    |      set up write code
2455  2035            ldd   tblpt | B
2456  4100            sti                 
2457  7774                  7774  |
2460  2036            ldd   eofpt | C
2461  4100            sti                 
2462  7775                  7775  |
2463  7500  wait      exf
2464  7040                  7040  |       select status dd65
2465  7600            ina         | D
2466  0202            lpn   2     |
2467  6404            zjb   wait  |       wait until keyboard 1 is hit
2470  7500            exf
2471  7140                  7140  |       select input from keyboard 1
2472  7600            ina         | E
2473  4204            stf   block |       set up block number arg
2474  0101            pta
2475  7030            jpi   rw    | F     jump to read/write routine
2476  0000  type
2477  0000  block
2500  2302            ldb   type  |
2501  6014            zjf   exit  | G     exit after write operation
2502  2200            ldf
2503  3000                  3000  |
2504  4013            std   size  |       set up size arg for print
2505  2100            ldi         | H
2506  7774                  7774  |
2507  4035            std   tblpt |       recover end of table pointer
2510  2100            ldi         |
2511  7775                  7775  |
2512  4036            std   eofpt |       recover eof pointer
2513  0101            pta
2514  7046            jpi   dis   | J
2515  7044  exit      jpi   consol
      0044  consol    equ   44
      0046  dis       equ   46
      0030  rw        equ   30
      0013  size      equ   13
      0036  eofpt     equ   36
      0035  tblpt     equ   35
      0000            end
```

```
      2555  blank   org   2555        subroutine to add letters bk
2555  0602          adn   2           in blank table to indicate
2556  4214          stf   link        blanking required
2557  2035          ldd   tblpt  |A    link made
2560  3213          adf   quart  |
2561  0704          sbn   4           addresses in blank table
2562  4077          std   temp        go to temporary storage
2563  2212  loop    ldf   letter |
2564  4177          sti   temp   |B
2565  0402          ldn   2      |
2566  5077          rad   temp   |
2567  4605          srf   flag   |C
2570  6605          pjb   loop   |
2571  7021          jpi   tb
2572  0000  link
2573  0400  quart         400
2574  5252  flag          5252
2575  6242  letter        6242
      0077  temp    equ   77
      0035  tblpt   equ   35
      0021  tb      equ   21
      0000          end
```

```
      2000  disply  org   2000          converts resistance and bias voltage
2000  0501          lcn   1             values to pot settings and makes up
2001  4056          std   flag          the dispay code for he file
2002  2200          ldf                 flag indicates whether info transfer
2003  3100                3100          is outline,pos or negative component
2004  4050          std   outlin        code transfer
2005  4026          std   temp      A
2006  2200          ldf
2007  3145                3145
2010  4052          std   end           end contains the end of file being
2011  2201          ldf   file          transferred
2012  4000  file            4000
2013  4036          std   eofpt         eofpt set at start of display file
2014  2126  trans   ldi   temp          transfer one block of code
2015  4136          sti   eofpt
2016  5436          aod   eofpt     B
2017  5426          aod   temp
2020  1452          scd   end
2021  6505          nzb   trans
2022  2056          ldd   flag          one block has been transferred
2023  6237          pjf   pos       C
2024  6304          njf   intrim
2025  3200  posout        3200
2026  3247  posend        3247
2027  0777  mask          777
2030  5456  intrim  aod   flag
2031  2152          ldi   end
2032  1720          scb   file
2033  4026          std   temp
2034  2200          ldf
2035  2720                2720
2036  4126          sti   temp
2037  2200          ldf
2040  7100                7100
2041  4051          std   rpos
2042  0602          adn   2
2043  4055          std   vpos          indices set for data tables
2044  2100          ldi
2045  1344                1344          load xyflag,determine if x or y calc
2046  6004          zjf   contin    D
2047  2200          ldf
2050  3020                3020
2051  4126          sti   temp          signifies a y calc
2052  2325  contin  ldb   posout
2053  4050          std   outlin
2054  2326          ldb   posend
2055  4052          std   end
2056  2152          ldi   end
2057  4054          std   x         E   x is ones complement of first x
```

```
2060  1331          lpb   mask      position of pos components in disply
2061  4013          std   exx       mask x to form dd65 code
2062  5452  pos     aod   end
2063  2057          ldd   pospt
2064  1451          scd   rpos   |G
2065  6075          zjf   skip   |G
2066  2152  four    ldi   end    |
2067  4076          std   addrss |
2070  2013          ldd   exx    |
2071  4176          sti   addrss |
2072  5452          aod   end    |H
2073  2361          ldb   file   |
2074  0110          ls3          |
2075  4363          stb   file   |
2076  6610          pjb   four   |
2077  6302          njf   twice
2100  6564  transl  nzb   trans
2101  0101  twice   pta
2102  7040          jpi   arith  |   compute pot settings
2103  0031                3l     |
2104  0051                rpos   |
2105  0002                vmax   |
2106  0026                temp   |
2107  0002                2      |
2110  0002                vmax   |
2111  0055                vpos   |
2112  0000                temp2  |
2113  0041                41     |
2114  0026                temp   |
2115  0000                temp2  |
2116  0000                temp2  |
2117  0041                41     |J
2120  0026                temp   |J
2121  0055                vpos   |
2122  0026                temp   |
2123  2055          ldd   vpos   |
2124  6103          nzf   3      |
2125  2376          ldb   mask   |
2126  4027          std   utemp  |   make temp very large
2127  0400          ldn   0      |
2130  4006          std   cntr   |
2131  2152  two     ldi   end    |
2132  4206          stf   outadr |
2133  5452          aod   end    |
2134  0101          pta          |
2135  7045          jpi   octdec |
2136  0026                temp   |
2137  0001                1      |
2140  0000  outadr              |
```

```
214   0000  outadr
2141  2001        ldd   utemp2   |
2142  4027        std   utemp     |
2143  2000        ldd   temp2     |  J
2144  4026        std   temp      |
2145  2406        lcd   cntr      |
2146  4006        std   cntr      |
2147  6516        nzb   two       |
2150  6003        zjf   more
2151  6567  posl  nzb   pos
2152  6552  trans2 nzb  transl
2153  0477  more  ldn   77        |
2154  5054        rad   x         |  K
2155  1256        lpf   maskl     |
2156  4013        std   exx       |
2157  2155  duo   lda   vpos      |
2160  5200        adf             |
2161  0400              400       |
2162  6025  skip  zjf   negset    |
2163  4026        std   temp      |  L
2164  2152        ldi   end       |
2165  4076        std   addrss    |
2166  2126        ldi   temp      |
2167  4176        sti   addrss    |
2170  5452        aod   end       |
2171  2406        lcd   cntr
2172  4006        std   cntr
2173  6514        nzb   duo
2174  0511        lcn   11        |
2175  5052        rad   end       |
2176  0404        ldn   4         |
2177  5055        rad   vpos      |  M
2200  0404        ldn   4         |
2201  5051        rad   rpos      |
2202  2050        ldd   outlin    |
2203  4026        std   temp      |
2204  6532        nzb   trans2    |
2205  5456  negset aod   flag
2206  0301        scn   1         |  N
2207  6002        zjf   neg       |
2210  6124        nzf   exit
2211  2221  neg   ldf   negend
2212  4052        std   end       |
2213  2216        ldf   negout    |  O
2214  4050        std   outlin    |
2215  2152        ldi   end       |
2216  4054        std   x         |  P
2217  1214        lpf   maskl     |
2220  4013        std   exx       |
```

```
2221  2053          ldd   negpt  | P
2222  4057          std   pospt  |
2223  2200          ldf
2224  7200          .     7200
2225  4051          std   rpos
2226  0602          adn   2
2227  4055          std   vpos
2230  6557          nzb   posl
2231  3300  negout         3300
2232  3347  negend        3347
2233  0777  maskl         777
2234  0101  exit    pta
2235  7041          jpi   trkbal | Q
2236  0101          pta
2237  7046          jpi   dis
2240  7021          jpi   tb
      0036  eofpt   equ   36
      0050  outlin  equ   50
      0051  rpos    equ   51
      0052  end     equ   52
      0054  x       equ   54
      0055  vpos    equ   55
      0056  flag    equ   56
      0013  exx     equ   13
      0000  temp2   equ   0
      0001  utemp2  equ   1
      0006  cntr    equ   6
      0076  addrss  equ   76
      0057  pospt   equ   57
      0053  negpt   equ   53
      0026  temp    equ   26
      0027  utemp   equ   27
      0046  dis     equ   46
      0021  tb      equ   21
      0041  trkbal  equ   41
      0040  arith   equ   40
      0002  vmax    equ   2
      0045  octdec  equ   45
      0000          end
```

```
      2517              org   2517      |     read/write routine
2517  0602              adn   2         |
2520  4076  rw          std   type      | A   first arg is a code determining
2521  0601              adn   1         |     type of operation-read or write
2522  4075              std   block     |     second arg is block number
2523  0601              adn   1         |
2524  4074              std   link      |     form return linkage
2525  7500              exf             |
2526  1161                    1161      | B   rewind to load
2527  2575              lci   block     |     set cntr to neg of
2530  4077              std   cntr      | C   block number
2531  7507  fwd         exf   read      |
2532  7600              ina             | D
2533  5477              aod   cntr      | E
2534  6703              njb   fwd       | F   pass over blocks
2535  2176              ldi   type      |
2536  6006              zjf   write     | G   type equal to zero means write
2537  7500              exf             |
2540  2131  read              2131      |
2541  7210              inp   four      | J
2542  7776                    7776      |
2543  7074              jpi   link      |
2544  7500  write       exf             |
2545  2111                    2111      |
2546  7303              out   four      | H
2547  7776                    7776      |
2550  7074              jpi   link
2551  4000  four              4000
      0077  cntr        equ   77
      0076  type        equ   76
      0075  block       equ   75
      0074  link        equ   74
      0000              end
```

```
      2600  octdec  org   2600        converts octal double precision sequence
2600  0602          adn   2       |   to bcd
2601  4061          std   adrnos  |   obtain arguments
2602  0601          adn   1       |
2603  4062          std   numnos  | A
2604  0601          adn   1       |
2605  4060          std   outadr  |
2606  0601          adn   1       |
2607  4004          std   temp2   |
2610  0413          ldn   buf
2611  0703          sbn   3
2612  4063          std   bufcon
2613  2216  limit   ldf   in      |   load word infin in output buffer
2614  4163          sti   bufcon  |
2615  5702          aob   limit   |
2616  5463          aod   bufcon  | B
2617  4611          srf   three   |
2620  6605          pjb   limit   |
2621  0503          lcn   3
2622  5307          rab   limit
2623  0101          pta           |
2624  0623          adn   23      | C
2625  4237          stf   last    |
2626  6223          pjf   start
2627  6322          njf   start
2630  4444  three         4444
2631  7145  in            7145        infin stands for infinity
2632  6671  fi            6671
2633  4520  n             4520
2634  0001  octbl         1       |
2635  0000                0       |
2636  0012                12      |
2637  0000                0       |
2640  0144                144     |
2641  0000                0     D |
2642  1750                1750    |
2643  0000                0       |
2644  3420                3420    |
2645  0002                2       |
2646  3240                3240    |
2647  0030                30      |
2650  0000  ul
2651  2161  start   ldi   adrnos      store arguments
2652  4061          std   adrnos  |
2653  2162          ldi   numnos  |
2654  4062          std   numnos  | E
2655  2160          ldi   outadr  |
```

```
2656  4060          std   outadr
2657  2004          ldd   temp2
2660  4277          stf   link
2661  0503  loop1    lcn   3          outer loop starts conversion of new word
2662  4064          std   wrdctr
2663  2200          ldf
2664  0000  last                     F
2665  4065          std   pwrl0
2666  2161          ldi   adrnos
2667  4066          std   num
2670  5461          aod   adrnos     G
2671  2161          ldi   adrnos
2672  4067          std   unum
2673  2064  loop    ldd   wrdctr     inner loop divides num by powers of ten
2674  0613          adn   buf
2675  4063          std   bufcon     H  buffer control set to oresent buffer location
2676  0101          pta
2677  7040          jpi   arith
2700  0041                41
2701  0066                num
2702  0065                pwrl0
2703  0014                code
2704  0031                31
2705  0014                code       J
2706  0065                pwrl0
2707  0016                temp6
2710  0002                2
2711  0066                num
2712  0016                temp6
2713  0066                num
2714  0502          lcn   2
2715  5065          rad   pwrl0
2716  2014          ldd   code       if code zero must alter code
2717  6005          zjf   fix        K
2720  1240          lpf   mask       L  mask off sign bit-numbers should be positive
2721  0712          sbn   12
2722  6222          pjf   output     M  indicates number too large
2723  6303          njf   okay
2724  0412  fix     ldn   12         N  enter zero code in code
2725  4014          std   code
2726  2756  okay    lcb   ul
2727  4357          stb   ul         O
2730  6005          zjf   lwr
```

```
2731  2014          ldd   code
2732  0111          ls6          |P
2733  4163          sti   bufcon |Q
2734  6541          nzb   loop   |
2735  2014  lwr     ldd   code   |
2736  1563          sci   bufcon |R
2737  4163          sti   bufcon |
2740  5464          aod   wrdctr S
2741  6746          njb   loop   T check if three words have been transmitted to bu
2742  0502          lcn   2      |
2743  5063          rad   bufcon |
2744  2163  output    ldi   bufcon   output buffer storage so output address
2745  4160          sti   outadr |U
2746  5463          aod   bufcon |
2747  5460          aod   outadr |
2750  4611          srf   cycle  |
2751  6705          njb   output |
2752  5461          aod   adrnos
2753  0501          lcn   1      |
2754  5062          rad   nummos |V
2755  6574          nzb   loop1  |
2756  7101          jfi   link
2757  0000  link
2760  3777  mask            3777
2761  3567  cycle           3567
      0013  buf     equ     13
      0060  outadr  equ     60
      0061  adrnos  equ     61
      0062  numnos  equ     62
      0063  bufcon  equ     63
      0064  wrdctr  equ     64
      0065  pwr10   equ     65
      0066  num     equ     66
      0067  unum    equ     67
      0040  arith   equ     40
      0004  temp2   equ     4
      0014  code    equ     14
      0016  temp6   equ     16
      0076  cntr    equ     76
      0000          end
```

```
     3220  sroot   org   3220          finds square root of nos. of range
3220 0602          adn   2       | A   1 to 4,000,000 octal
3221 4006          std   link    |     make link
3222 0577          lcn   77
3223 4055          std   index5  | B   bound number of iterations
3224 0400          ldn   0
3225 4065          std   ubuf4
3226 4037          std   utwo
3227 0402          ldn   2
3230 4036          std   two     | C
3231 4064          std   buf4    |
3232 2061          ldd   usqare  | D   check if square greater than 3777
3233 6004          zjf   start   |
3234 2200          ldf
3235 2000                2000    |
3236 4064          std   buf4    | E
3237 0101  start   pta
3240 7040          jpi   arith   |
3241 0041                41      |
3242 0060                square  | F
3243 0064                buf4    |
3244 0015                tpt     |
3245 0101  root    pta
3246 7040          jpi   arith   |
3247 0031                31      |
3250 0015                tpt     |   comute estimated root squared r2
3251 0015                tpt     |
3252 0064                buf4    |
3253 0002                2       |
3254 0060                square  |   form square-r2
3255 0064                buf4    | G
3256 0064                buf4    |
3257 0041                41      |
3260 0064                buf4    |   form square-rt2 over 2timesr
3261 0015                tpt     |
3262 0066                buf6    |
3263 2066          ldd   buf6    |   buf6 scaled up by two
3264 6206          pjf   pos     |
3265 0501          lcn   1       |
3266 4064          std   buf4    |
3267 0500          lcn   0       | H
3270 4065          std   ubuf4   |
3271 6305          njf   scal    |
3272 0201  pos     lpn   1       |
3273 4064          std   buf4    |   obtain round off error
```

```
3274  0400          ldn   0
3275  4065          std   ubuf4
3276  0101  scal    pta
3277  7040          jpi   arith      |
3300  0041                41         |
3301  0066                buf6       | J
3302  0036                two        |
3303  0066                buf6       |
3304  0001                1          |   add round off correction
3305  0064                buf4       |
3306  0066                buf6       | K
3307  0066                buf6       |
3310  2066          ldd   buf6
3311  3067          add   ubuf6      |   sense if correction factor zero
3312  6011          zjf   exit       | L
3313  0101          pta
3314  7040          jpi   arith      |
3315  0001                1          |
3316  0015                tpt        | M
3317  0066                buf6       |
3320  0015                tpt        |   add correction factor
3321  5455          aod   index5     | N
3322  6555          nzb   root       | O  check if iteration bound reached
3323  7006  exit    jpi   link
      0006  link    equ   6
      0040  arith   equ   40
      0015  tpt     equ   15
      0064  buf4    equ   64
      0065  ubuf4   equ   65
      0060  square  equ   60
      0061  usqare  equ   61
      0036  two     equ   36
      0037  utwo    equ   37
      0066  buf6    equ   66
      0067  ubuf6   equ   67
      0055  index5  equ   55
      0000          end
```

```
3100    2000 this is the main outlin code
3101    0440
3102    0240
3103    7664
3104    5000
3105    7700
3106    2000
3107    0271
3110    0240
3111    7650
3112    5600
3113    7700
3114    2000
3115    0271
3116    0537
3117    7664
3120    5400
3121    7700
3122    2000
3123    0440
3124    0000
3125    7665
3126    5051
3127    5757
3130    5150
3131    1256
3132    5612
3133    0002
3134    5600
3135    7700
3136    0000
3137    0737
3140    0320
3141    0000
3142    6361
3143    4363
3144    7700
3145    0041
.
```

```
3200    2000 this is the code for the pos
3201    0000 outline
3202    0240
3203    7605
3204    5614
3205    5050
3206    5553
3207    1650
3210    5014
3211    7605
3212    5657
3213    5555
3214    5776
3215    0556
3216    7700
3217    2000
3220    0000
3221    0110
3222    5050
3223    4253
3224    5151
3225    5342
3226    7700
3227    0000
3230    0000
3231    0144
3232    0000
3233    0000
3234    0000
3235    0000
3236    7700
3237    0000
3240    0000
3241    0064
3242    0000
3243    0000
3244    0000
3245    0000
3246    7700
3247    7460
3250    3201
3251    3220
3252    3230
3253    3240
3254    3232
3255    3242
3256    3235
3257    3245
.
```

```
100       2000 this is the code for the neg
     0000       outline
     0537
     7605
     5250
     5355
     5012
     1450
     5014
     7605
     5253
     5151
     5376
     0552
     7700
     2000
     0000
     0667
     5050
     4657
     5555
     5746
     7700
     0000
     0000
     0653
     0000
     0000
     0000
     0000
     7700
     0000
     0000
     0677
     0000
     0000
     0000
     0000
     7700
     7460
     3301
     3320
     3330
     3340
     3332
     3342
     3335
     3345
.
```

```
      3400  fixdpt  org   3400
3400  0602          adn   2          |
3401  4006          std   link       | A   make link
3402  0400          ldn   0          |
3403  4037          std   utwo       |
3404  4014          std   scale      | B
3405  0402          ldn   2          |
3406  4036          std   two        |
3407  0101          pta              |
3410  7040          jpi   arith      |
3411  0031                3l         |
3412  0002                vmax       | C
3413  0006                sens       |
3414  0060                buf        |   compute vmax times sens
3415  2061  loop    ldd   ubuf       |
3416  1212          lpf   mask       |
3417  6112          nzf   next       | D
3420  0101          pta              |
3421  7040          jpi   arith      |
3422  0031                3l         |
3423  0060                buf        |
3424  0036                two        | E
3425  0060                buf        |   scale up buf
3426  5414          aod   scale      |
3427  6612          pjb   loop       |   unconditional jump
3430  7400  mask            7400
3431  0101  next    pta
3432  7040          jpi   arith      |
3433  0041                41         |
3434  0060                buf        |
3435  0012                dist       |
3436  0060                buf        |   compute vmax sens/dist
3437  0041                41         |
3440  0060                buf        |
3441  0004                diff       |
3442  0060                buf        | F
3443  0002                2          |
3444  0055                index5     |
3445  0027                dtl        |
3446  0026                dt         |
3447  0031                3l         |
3450  0060                buf        |
3451  0026                dt         |
3452  0060                buf        |   compute vmax sens dt/dist diff
3453  2061  loop1   ldd   ubuf       |   scale up buf
3454  6202          pjf   2          | G
3455  2461          lcd   ubuf       |
3456  1326          lpb   mask       |
3457  6111          nzf   contin     |
3460  0101          pta              |
3461  7040          jpi   arith      | H
3462  0031                3l         |
3463  0060                buf        |
```

```
3464  0036                    two
3465  0060                    buf      | H
3466  5414          aod       scale    |
3467  6614          pjb       loop1
3470  0101  contin  pta
3471  7040          jpi       arith    |
3472  0041                    41       |
3473  0060                    buf      |
3474  0012                    dist     |
3475  0064                    rt2      |   compute vmax sens dt/diff dist2
3476  0031                    31       |
3477  0064                    rt2      | J
3500  0055                    index5   |
3501  0064                    rt2      |   compute vmax sens dt t/diff dist2
3502  0002                    2        |
3503  0060                    buf      |
3504  0064                    rt2      |
3505  0064                    rt2      |   rt2 now computed
3506  0501          lcn       1
3507  5014          rad       scale    | K
3510  2014  loop2   ldd       scale    |
3511  6013          zjf       exit     | L
3512  0101          pta
3513  7040          jpi       arith
3514  0041                    41       |
3515  0064                    rt2      |
3516  0036                    two      | M
3517  0064                    rt2      |   scale rt2 back down
3520  0501          lcn       1        |
3521  5014          rad       scale    |
3522  0400          ldn       0        |
3523  6413          zjb       loop2    |
3524  2064  exit    ldd       rt2
3525  0201          lpn       1        |
3526  4060          std       buf      |
3527  0400          ldn       0        | N
3530  4061          std       ubuf     |   get round off error
3531  2064          ldd       rt2      |
3532  6203          pjf       corect   | P
3533  2460          lcd       buf      |
3534  4060          std       buf      | Q
3535  0101  corect  pta                |
3536  7040          jpi       arith    |
3537  0041                    41       |
3540  0064                    rt2      |
3541  0036                    two      |
3542  0064                    rt2      | R
3543  0001                    1        |
3544  0064                    rt2      |
3545  0060                    buf      |
3546  0064                    rt2      |   add round off correction
3547  7006          jpi       link     |
      0006  link    equ       6
      0014  scale   equ       14
```

```
0026  dt      equ   26
0027  dtl     equ   27
0036  two     equ   36
0037  utwo    equ   37
0002  vmax    equ   2
0040  arith   equ   40
0055  index5  equ   55
0060  buf     equ   60
0061  ubuf    equ   61
0000  sens    equ   0
0012  dist    equ   12
0064  rt2     equ   64
0065  urt2    equ   65
0004  diff    equ   4
0000          end
```

APPENDIX III

MAINTENANCE OF ACCURACY OF ARITHMETIC COMPUTATIONS

Section 5 of Chapter III refers to the use of a CDC 160 computer for the graphically orientated design process. This particular computer was limited in available memory capacity as well as computer word size. This limitation imposed the use of twenty-two bit fixed-point arithmetic for all arithmetic computations. In order to maintain accuracy, special procedures were necessary to avoid large round-off errors. By way of illustration of the method used, reference is made to figure 36 which is the flow diagram for the fixdpt subroutine. This subroutine does a special sequence of arithmetic operations which will be studied. Starting with the multiplication of step C, steps D and E are carried out to ensure the number is scaled up to as large a number as the twenty-two bit format will permit. The variable "scale" contains the power of two used in scaling up this product. Further operations are performed in F, and in G the number is scaled up again. Still further computations are performed using the full accuracy of the machine while choosing the order of operations to ensure overflow does not occur. In L and M, the result is scaled down until the number has a scale factor of two. The last bit indicates whether the number is to be rounded off by truncation only or by truncation with unit correction. For negative numbers the correction factor is negative as sensed in step Q. Descaling to unity scale factor and addition of round-off correction in step R completes the computations. The above procedure ensures accuracy without the use of floating-point arithmetic.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	Prof. Mitchell L. Cotton (Thesis Advisor) Department of Electrical Engineering Naval Postgraduate School, Monterey, California	5
4.	LT John M. C. TYNAN 2121 Trapani Circle Monterey, California 93940	1

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

A COMPUTER GRAPHICS APPROACH TO NON-LINEAR CIRCUIT DESIGN

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Master of Science Thesis

5. AUTHOR(S) (Last name, first name, initial)

TYNAN, John M. C., Lieutenant, Royal Canadian Navy

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
2 June 1967	121	3

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. AVAILABILITY/LIMITATION NOTICES

[redacted] is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the Naval Postgraduate School.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

This investigation is concerned with computer and programming requirements for on-line network design. A class of problem involving non-linear diode resistance networks was chosen. Such networks provide a useful form of general purpose function generator, here applied to the specific task of symbol generation. A procedure was implemented whereby the user "draws" the required symbol to be generated on a cathode ray tube display, and output is obtained in the form of a labeled circuit diagram of the synthesized network. The validity of the computer-graphic design program was demonstrated by construction and testing of a particular symbol generator.

DD FORM 1 JAN 64 1473

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Network design						
Computer graphics						
Function generation						
Symbol generation